

INTRODUCCIÓN

Saludos! En este documento PDF se describirá paso a paso el curso realizado en UDEMRY de DJANGO con capturas del proceso y explicándolas. DJANGO es un framework de desarrollo web de código abierto, escrito en Python, cuya meta fundamental es facilitar la creación de sitios web complejos.

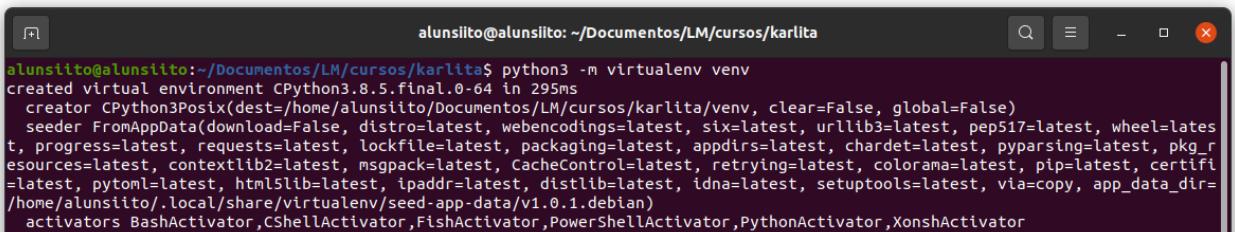
Para usar **DJANGO** necesitamos tener instalado lo siguiente:

- **Python** (En este caso será Python 3)
 - Instalado con: `$ sudo apt install python3`
- **PIP**
 - Instalado con: `$ sudo apt install python3-pip`
- **Virtualenv**
 - Instalado con: `$ pip3 install virtualenv`
- **Python-django**
 - Creamos un entorno virtual → `$ python3 -m virtualenv [nombre]`
 - Accedemos al entorno → `$ source [entorno]/bin/activate`
 - Instalamos django con: `$ pip3 install django {última version}`
- Un editor de código (en mi caso es sublime text 3)

En este curso utilizaremos un entorno virtual para instalar y hacer todas las modificaciones necesarias sin influir al host anfitrión.

Una vez tengamos todo listo... Empezamos!

1. INSTALACIÓN DE DJANGO

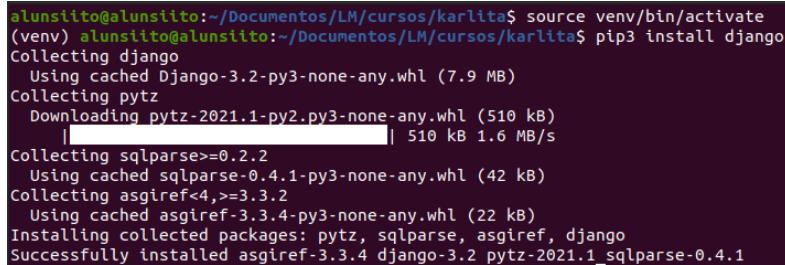


```
alunslito@alunslito:~/Documentos/LM/cursos/karlita$ python3 -m virtualenv venv
  created virtual environment CPython3.8.5.final.0-64 in 295ms
    creator CPython3Posix(dest=/home/alunslito/Documentos/LM/cursos/karlita/venv, clear=False, global=False)
    seeder FromAppData(download=False, distro=latest, webencodings=latest, six=latest, urllib3=latest, pep517=latest, wheel=latest, progress=latest, requests=latest, lockfile=latest, packaging=latest, appdirs=latest, chardet=latest, pyParsing=latest, pkg_resources=latest, contextlib2=latest, msgpack=latest, CacheControl=latest, retrying=latest, colorama=latest, pip=latest, certifi=latest, pytoml=latest, htmlslib=latest, ipaddr=latest, distlib=latest, idna=latest, setuptools=latest, via=copy, app_data_dir=/home/alunslito/.local/share/virtualenv/seed-app-data/v1.0.1.debian)
    activators BashActivator,CShellActivator,FishActivator,PowerShellActivator,XonshActivator
```

En primer lugar, empezaremos con la creación del entorno virtual, para ello, escribiremos el comando mencionado anteriormente:

```
$ python3 -m virtualenv [nombre]
```

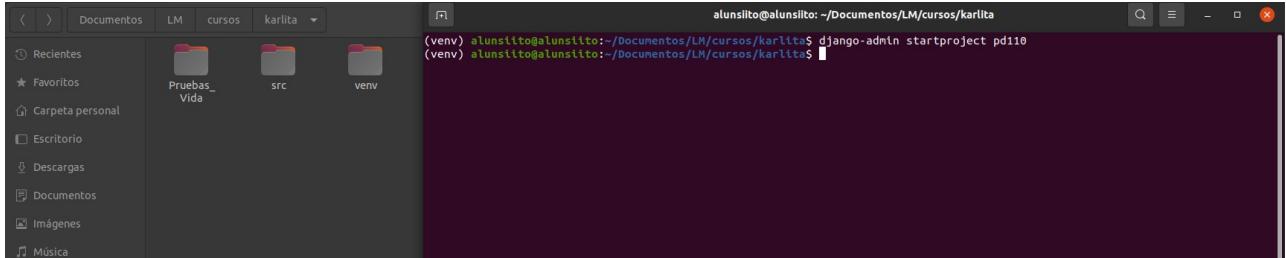
Una vez tengamos nuestro entorno listo, vamos a iniciararlo y a instalar *django* como expliqué en la introducción.



```
alunslito@alunslito:~/Documentos/LM/cursos/karlita$ source venv/bin/activate
(venv) alunslito@alunslito:~/Documentos/LM/cursos/karlita$ pip3 install django
Collecting django
  Using cached Django-3.2-py3-none-any.whl (7.9 MB)
Collecting pytz
  Downloading pytz-2021.1-py2.py3-none-any.whl (510 kB)
    |██████████| 510 kB 1.6 MB/s
Collecting sqlparse>=0.2.2
  Using cached sqlparse-0.4.1-py3-none-any.whl (42 kB)
Collecting asgiref<4,>=3.3.2
  Using cached asgiref-3.3.4-py3-none-any.whl (22 kB)
Installing collected packages: pytz, sqlparse, asgiref, django
Successfully installed asgiref-3.3.4 django-3.2 pytz-2021.1 sqlparse-0.4.1
```

E instalamos *django* con *pip3*.

2. NUEVO PROYECTO



A continuación, creamos nuestro proyecto con el comando *django-admin* (karlita usa la ruta absoluta para el comando ya que no tiene la variable de entorno establecida). El proyecto lo llamaremos **pd110**. Este proyecto se alojará dentro de la carpeta *src*, la cual originalmente se llama como el proyecto pero, al crear dos iguales una dentro de otra, le cambié el nombre para diferenciarlos a “*src*”.

```
alunslito@alunslito:~/Documentos/LM/cursos/karlita/src
(venv) alunslito@alunslito:~/Documentos/LM/cursos/karlita$ django-admin startproject pd110
(venv) alunslito@alunslito:~/Documentos/LM/cursos/karlita$ python3 manage.py runserver
python3: can't open file 'manage.py': [Errno 2] No such file or directory
(venv) alunslito@alunslito:~/Documentos/LM/cursos/karlita$ cd src/
(venv) alunslito@alunslito:~/Documentos/LM/cursos/karlita/src$ python3 manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

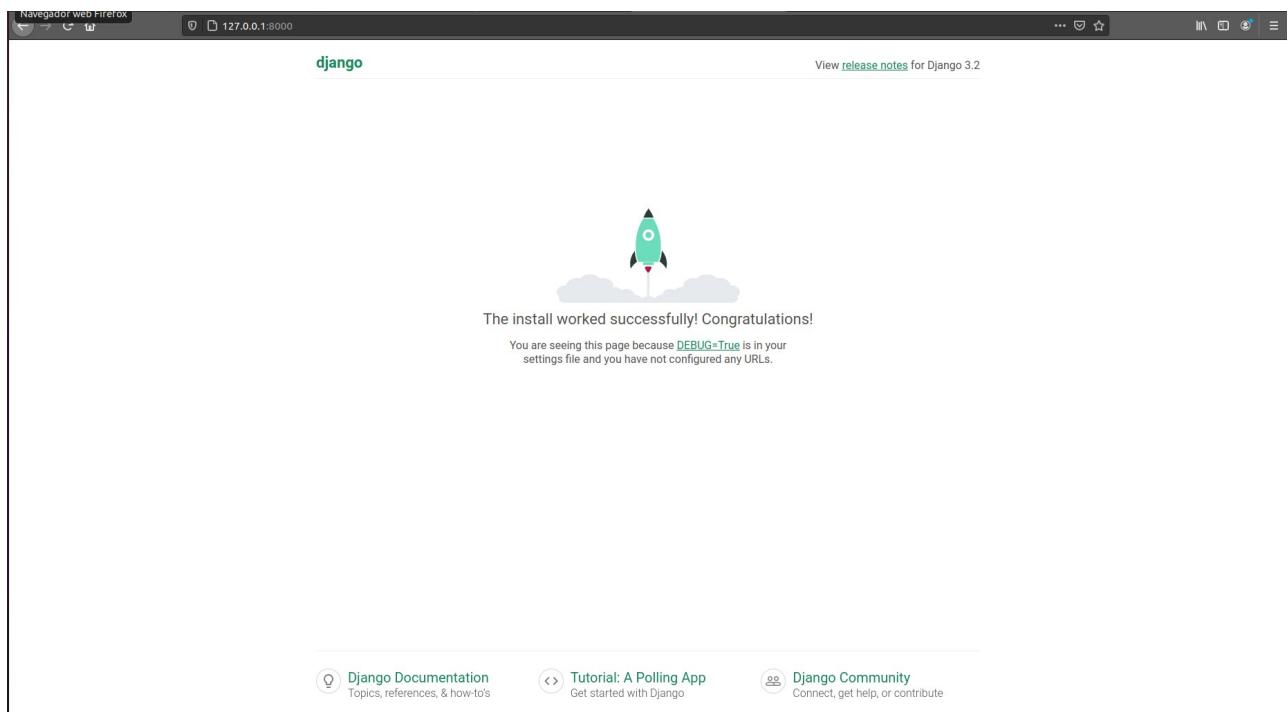
You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
April 09, 2021 - 08:54:01
Django version 3.2, using settings 'pd110.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
[09/Apr/2021 08:54:37] "GET / HTTP/1.1" 200 10697
[09/Apr/2021 08:54:37] "GET /static/admin/css/fonts.css HTTP/1.1" 200 423
[09/Apr/2021 08:54:37] "GET /static/admin/fonts/Roboto-Bold-webfont.woff HTTP/1.1" 200 86184
[09/Apr/2021 08:54:37] "GET /static/admin/fonts/Roboto-Regular-webfont.woff HTTP/1.1" 200 85876
Not Found: /favicon.ico
[09/Apr/2021 08:54:37] "GET /favicon.ico HTTP/1.1" 404 1994
[09/Apr/2021 08:54:37] "GET /static/admin/fonts/Roboto-Light-webfont.woff HTTP/1.1" 200 85692
```

Una vez lo tengamos listo, ejecutamos nuestra aplicación con:

\$ *python3 manage.py runserver*

(Nosotros usamos Python 3, karlita usa python 2)

La vista previa de nuestra aplicación la podremos ver en nuestro navegador si accedemos al puerto 8000 de nuestro *localhost*:



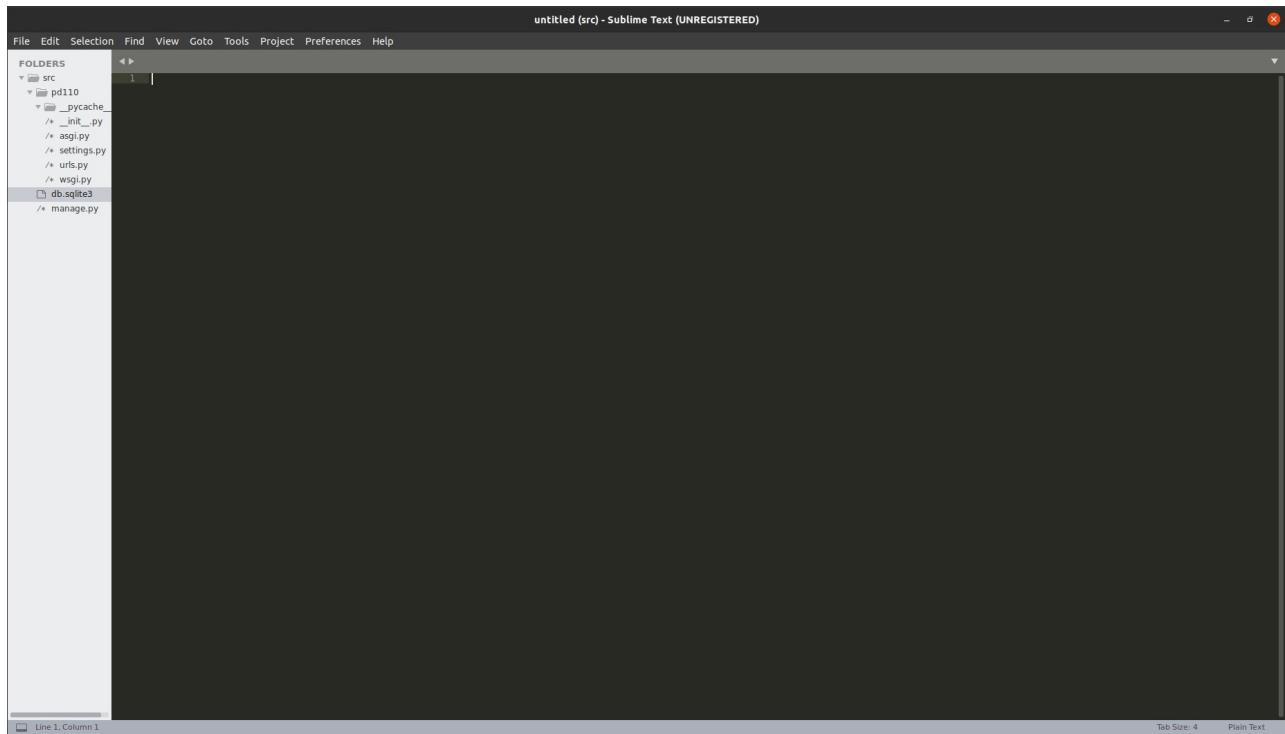
Y como podemos observar, nuestro proyecto funciona perfectamente y está listo para ser configurado.

3. PRIMERA MIGRACIÓN

En este capítulo, vamos a ver como hacer una migración de los datos de nuestra aplicación a la base de datos. Para ello escribiremos lo siguiente:

```
C(venv) alunslito@alunslito:~/Documentos/LM/cursos/karlita/src$ python3 manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying sessions.0001_initial... OK
(venv) alunslito@alunslito:~/Documentos/LM/cursos/karlita/src$
```

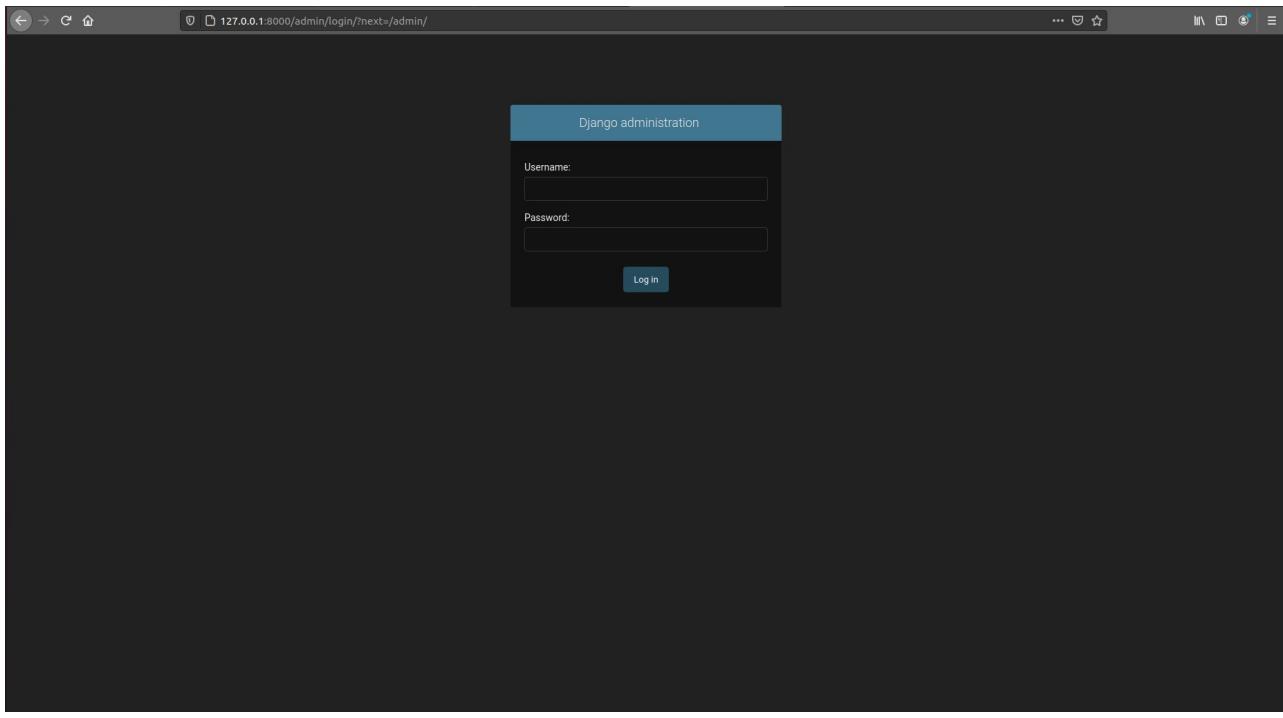
Así todo el contenido de la aplicación se ha actualizado.



Para seguir con los posteriores capítulos abriremos en nuestro editor la carpeta de nuestro proyecto.

4. CREACIÓN DE UN SUPERUSUARIO Y ADMINISTRACIÓN DEL ADMIN SITE

En este capítulo procederemos a crear un *superusuario* para poder administrar nuestra aplicación desde el *admin site*.



Aquí es donde iniciaremos sesión con el *superusuario*. Como no disponemos de él, vamos a crearlo. Nos vamos a nuestra terminal y escribimos este comando:

```
^C(venv) alunsiito@alunsiito:~/Documentos/LM/cursos/karlita/src$ python3 manage.py createsuperuser
Username (leave blank to use 'alunsiito'): miguelgulu
Email address: miguel.guerrero.luna.alu@iesfernandoaguilar.es
Password:
Password (again):
The password is too similar to the username.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.
(venv) alunsiito@alunsiito:~/Documentos/LM/cursos/karlita/src$
```

Nos vendrán unos campos a llenar para crearlo que habrá que completar...

Una vez creado, nos vamos al *admin site* e iniciamos sesión con el mismo pero, antes, vamos a cambiar el idioma del sitio (ya que de forma predeterminada esta en inglés):

The screenshot shows a Sublime Text window with the following details:

- File Path:** ~/Documentos/LM/cursos/karlita/src/pd110/settings.py (src) - Sublime Text (UNREGISTERED)
- File List:** The sidebar shows the project structure: FOLDERS (src, pd110), __pycache__, __init__.py, asgi.py, settings.py, urls.py, wsgi.py, and db.sqlite3.
- Code Editor:** The main pane displays the `settings.py` file content. The code includes configuration for database, authentication validators, internationalization, static files, and default fields. A cursor is visible at line 106, where the `LANGUAGE_CODE` variable is defined.
- Right Panel:** A sidebar panel on the right contains a tree view of the project's file structure and a preview area showing the current file's content.

```
File Edit Selection Find View Goto Tools Project Preferences Help

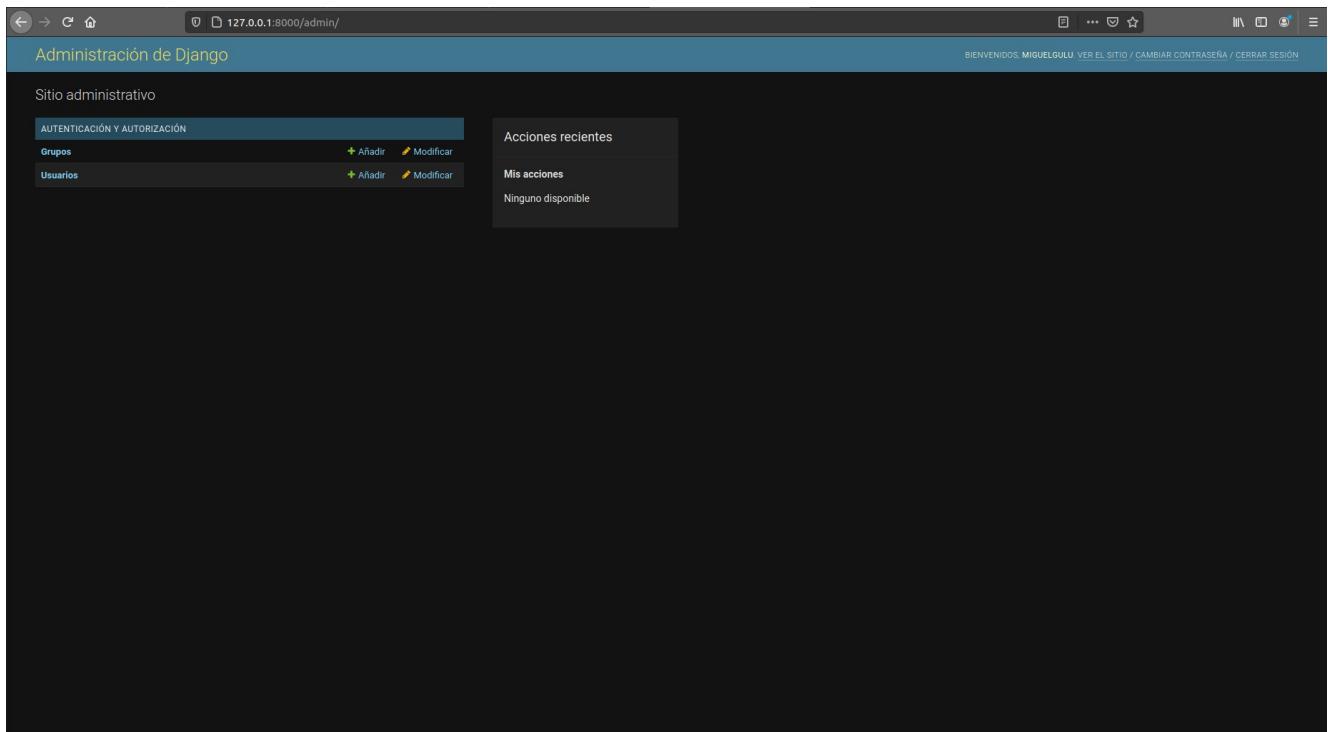
FOLDERS
src
pd110
__pycache__
__init__.py
asgi.py
settings.py
urls.py
wsgi.py
db.sqlite3
manage.py

settings.py x
79     'NAME': BASE_DIR / 'db.sqlite3',
80 }
81
82
83
84 # Password validation
85 # https://docs.djangoproject.com/en/3.2/ref/settings/#auth-password-validators
86
87 AUTH_PASSWORD_VALIDATORS = [
88     {
89         'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
90     },
91     {
92         'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
93     },
94     {
95         'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
96     },
97     {
98         'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
99     },
100 ]
101
102
103 # Internationalization
104 # https://docs.djangoproject.com/en/3.2/topics/i18n/
105 LANGUAGE_CODE = 'es'
106 TIME_ZONE = 'UTC'
107 USE_I18N = True
108 USE_L10N = True
109 USE_TZ = True
110
111
112 # Static files (CSS, JavaScript, Images)
113 # https://docs.djangoproject.com/en/3.2/howto/static-files/
114 STATIC_URL = '/static/'
115
116
117 # Default primary key field type
118 # https://docs.djangoproject.com/en/3.2/ref/settings/#default-auto-field
119
120 DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'

Line 106, Column 20
Spaces: 4
Python
```

En el archivo de “*settings.py*” en la línea de **LANGUAGE_CODE** escribiremos como idioma “es”.

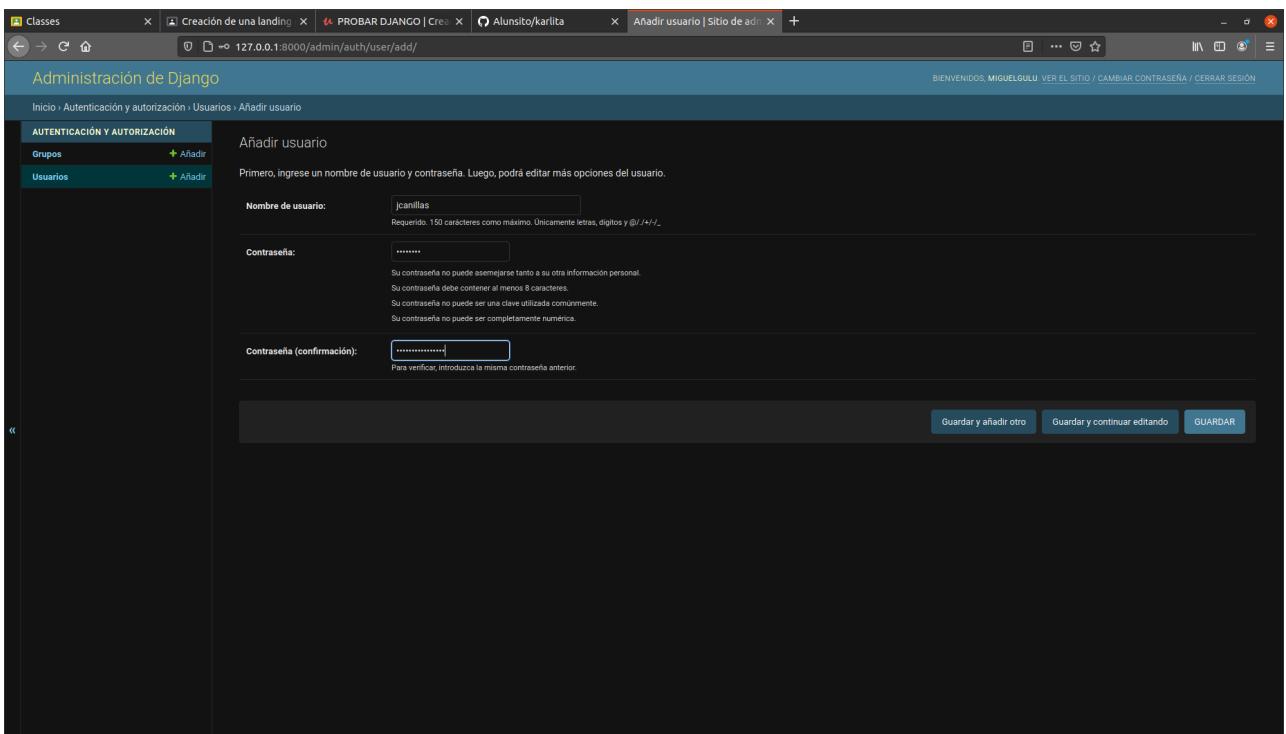
Guardamos cambios y procederemos a abrir la página.



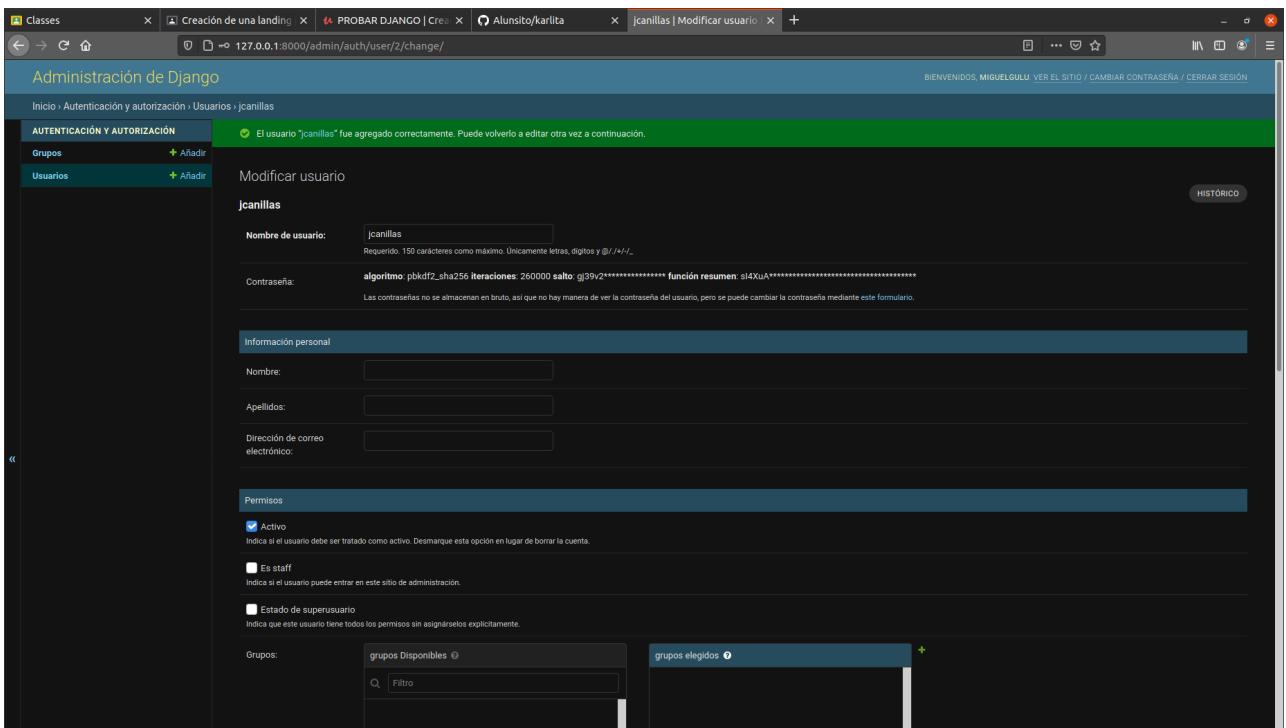
Como podemos comprobar, en nuestro *admin site* tenemos por ahora los grupos y usuarios que podremos configurar. Vamos a crear un usuario.

A screenshot of the Django Admin site users list page. The URL in the address bar is 127.0.0.1:8000/admin/auth/user/. The page shows a table with one user listed: "miguelgulu" (miguel.guerrero.luna.alu@iesfernandoaagular.es). The user is marked as a staff member ("ES STAFF"). The table has columns: "NOMBRE DE USUARIO", "DIRECCIÓN DE CORREO ELECTRÓNICO", "NOMBRE", "APELLIDOS", and "ES STAFF". To the left of the table is a search bar and a "FILTRO" sidebar with options for "Por es staff", "Por estado de superusuario", and "Por activo".

Aquí nos saldrán todos los usuarios que hayamos creado. La primera vez que entremos solamente veremos el *superusuario* que creamos. Vamos a crear el nuevo manualmente:



Rellenamos todos los campos que nos pone en pantalla y le damos a guardar.



Y finalmente nos dirá que se ha creado el usuario con éxito.

5. PRIMERA APLICACIÓN

Para crear nuestra primera aplicación debemos ejecutar el comando correspondiente en nuestro terminal. Usaremos como nombre de aplicación “**boletín**”:

```
(venv) alunslito@alunslito:~/Documentos/LM/cursos/karlita/src$ python3 manage.py startapp boletin
(venv) alunslito@alunslito:~/Documentos/LM/cursos/karlita/src$
```

A continuación, tendremos que añadir la aplicación a nuestro proyecto. Para ello, iremos al archivo de settings y en la parte de **INSTALLED_APPS** añadiremos nuestra aplicación. A diferencia del curso, karlita pone directamente el nombre de la aplicación pero como nuestra versión es mucho más reciente debemos disponerla de otra forma. Nos tenemos que dirigir al archivo *apps.py* y veremos una función con un nombre. Hay que tener en cuenta el nombre de la misma para usarlo en el siguiente paso.



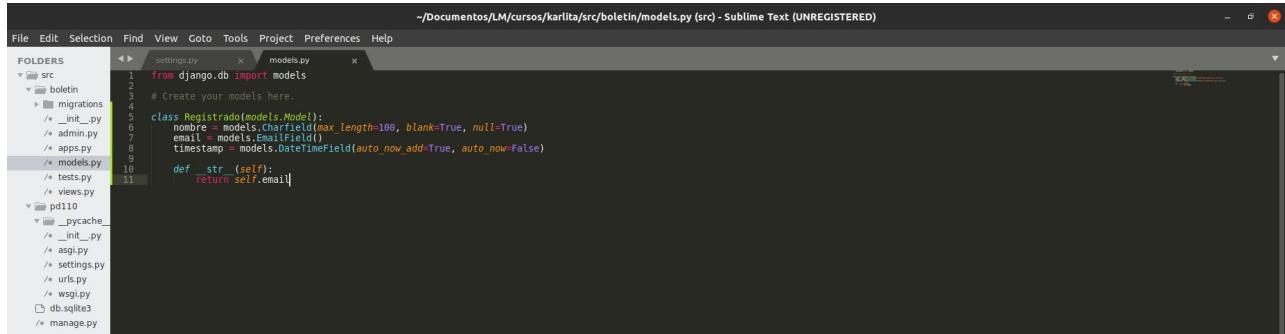
Ahora sí, vamos al archivo de settings y buscamos la sección mencionada anteriormente. Karlita pone directamente el nombre pero, nosotros, tendremos que escribir ‘[nombre_app].apps.[función]’

donde hay que escribir el nombre de la app (**boletin**) y la función que mencionamos anteriormente. Sería tal que así:

```
/* wsgi.py
db.sqlite3
/* manage.py
      30     # Application definition
      31     INSTALLED_APPS = [
      32         'django.contrib.admin',
      33         'django.contrib.auth',
      34         'django.contrib.contenttypes',
      35         'django.contrib.sessions',
      36         'django.contrib.messages',
      37         'django.contrib.staticfiles',
      38         'boletin.apps.BoletinConfig',
      39     ]
      40
      41 ]
```

6. PRIMER MODELO

Para crear nuestro primer modelo debemos crear una clase. Abrimos nuestro archivo *models.py* y crearemos una clase como la siguiente:



```

File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS
src
└── boletin
    ├── migrations
    │   └── __init__.py
    ├── admin.py
    ├── apps.py
    ├── models.py
    ├── tests.py
    └── views.py
└── pd110
    ├── __pycache__
    │   ├── __init__.py
    │   ├── asgi.py
    │   ├── settings.py
    │   ├── urls.py
    │   └── wsgi.py
    └── db.sqlite3
    └── manage.py

1  from django.db import models
2
3  # Create your models here.
4
5  class Registrado(models.Model):
6      nombre = models.CharField(max_length=100, blank=True, null=True)
7      email = models.EmailField()
8      timestamp = models.DateTimeField(auto_now=True, auto_now=False)
9
10     def __str__(self):
11         return self.email

```

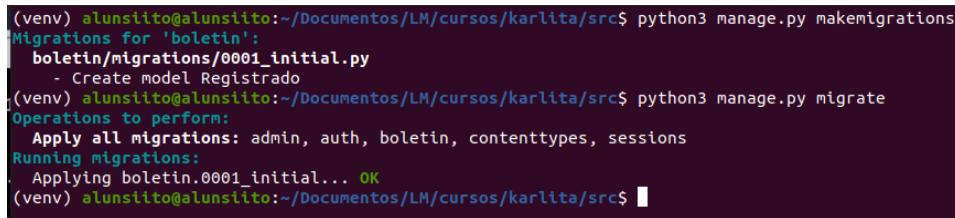
Nosotros al final para declarar lo que va a devolver la clase usamos “`__str__`” porque usamos **Python 3**, pero si fuera **Python 2** seria “`__unicode__`”.

Nuestra clase tiene un nombre, un *email* y una fecha de creación, cada uno con su campo tipo *char*, *email* y *datetime* respectivamente.

Finalmente para subir la clase a la base de datos debemos hacer:

`$ python3 manage.py makemigrations` → Para guardar los cambios

`$ python3 manage.py migrate` → Para subir los cambios a la base



```

(venv) alunsiito@alunsiito:~/Documentos/LM/cursos/karlita/src$ python3 manage.py makemigrations
Migrations for 'boletin':
  - Create model Registrado
(venv) alunsiito@alunsiito:~/Documentos/LM/cursos/karlita/src$ python3 manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, boletin, contenttypes, sessions
Running migrations:
  Applying boletin.0001_initial... OK
(venv) alunsiito@alunsiito:~/Documentos/LM/cursos/karlita/src$ 

```

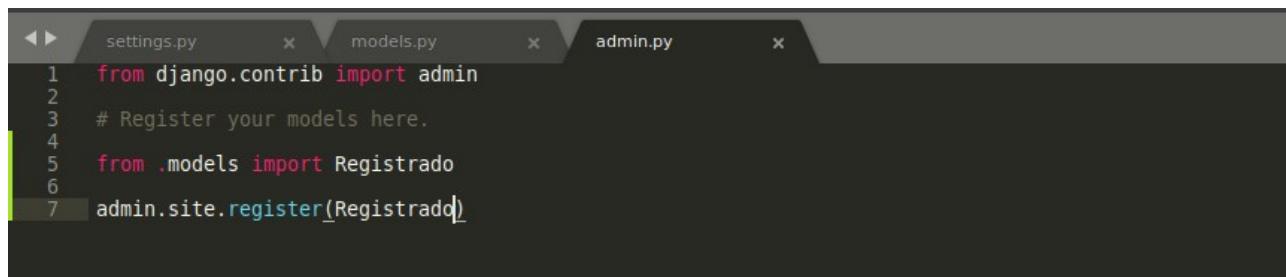
7. REGISTROS DE MODELOS A TRAVÉS DEL SHELL

Una vez tenemos registrado nuestro modelo, vamos a añadir registros al mismo. Hay dos maneras: manualmente mediante el *admin* o mediante un *shell* de *python* que ofrece el mismo proyecto de django. En este capítulo vamos a hacerlo mediante el *shell*. Este *shell* se abre con el comando siguiente:

```
(venv) alunsiito@alunsiito:~/Documentos/LM/cursos/karlita/src$ python3 manage.py shell
Python 3.8.5 (default, Jan 27 2021, 15:41:15)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> from boletin.models import Registrado
>>> gente = Registrado.objects.all()
>>> gente
<QuerySet []>
>>> persona1 = Registrado.objects.create(nombre='pepito', email='palotes@gmail.com')
>>> persona1
<Registrado: palotes@gmail.com>
>>> gente
<QuerySet [<Registrado: palotes@gmail.com>]>
>>> 
```

Para añadir los registros, hay que importar el modelo creado. Una vez importado podemos añadir registros al mismo como se ve en pantalla. El conjunto de registros se ve como una lista como se puede comprobar en el **QuerySet**. Para añadir un registro hay que completar todos los campos que componen el modelo.

Esta es una manera muy simple de añadir nuevos registros a nuestros modelos. Como mencioné anteriormente, también se puede hacer por el *admin site*. Para ello, vamos a añadir nuestro modelo al mismo de esta forma:



```
settings.py      models.py      admin.py
1  from django.contrib import admin
2
3  # Register your models here.
4
5  from .models import Registrado
6
7  admin.site.register(Registrado)
```

Una vez iniciado sesión en nuestro *admin site*, veremos que en modelos tenemos Registrado. Una vez entramos veremos el registro que introducimos anteriormente mediante el shell:

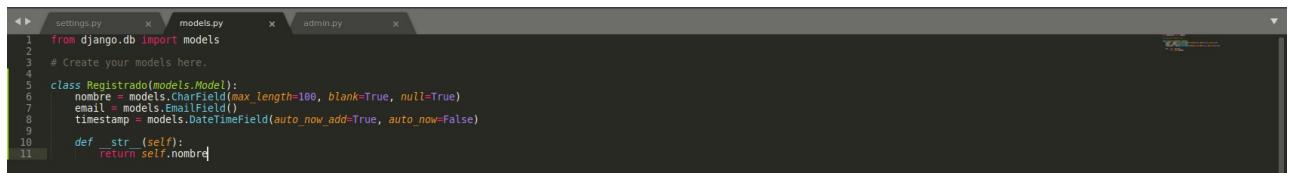
The screenshot shows the Django Admin interface with a dark theme. On the left, there's a sidebar with 'AUTENTICACIÓN Y AUTORIZACIÓN' and 'BOLETIN' sections. Under 'BOLETIN', there's a 'Registrados' model with two entries: 'jcanillas' and 'Usuario'. Each entry has a green '+' icon for 'Añadir' and a blue pencil icon for 'Modificar'.

This screenshot shows the 'Modificar registrado' page for the 'pepito' user. The user information is displayed: Nombre: pepito and Email: palotes@gmail.com. Below the form are buttons for 'Eliminar', 'Guardar y añadir otro', 'Guardar y continuar editando', and 'GUARDAR'.

Si hacemos click en el email podremos modificar el registro:

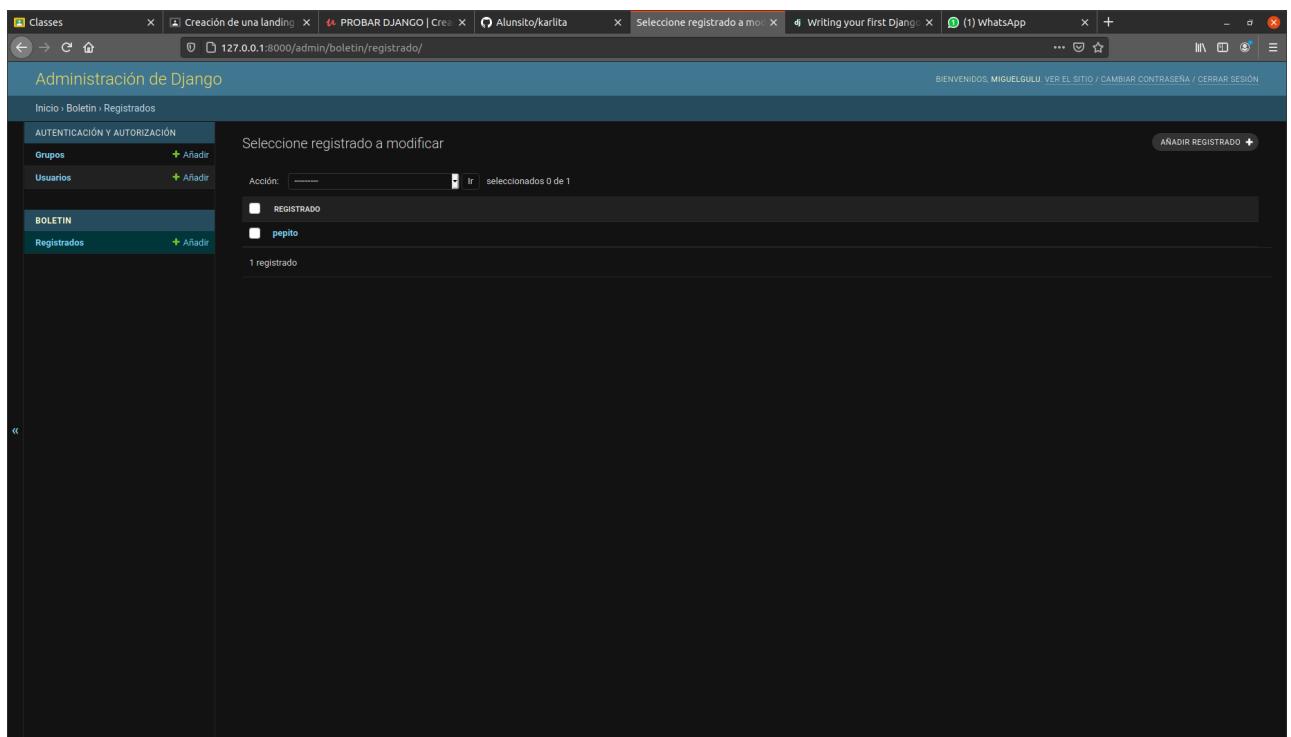
This screenshot shows the 'Modificar registrado' page for the 'pepito' user. The 'Email' field is highlighted with a red border, indicating it is selected or being edited. The other fields ('Nombre') are also present.

En **Registrados** nos sale solamente el email, pero podemos pedirle que nos devuelva por ejemplo el nombre de la siguiente manera:

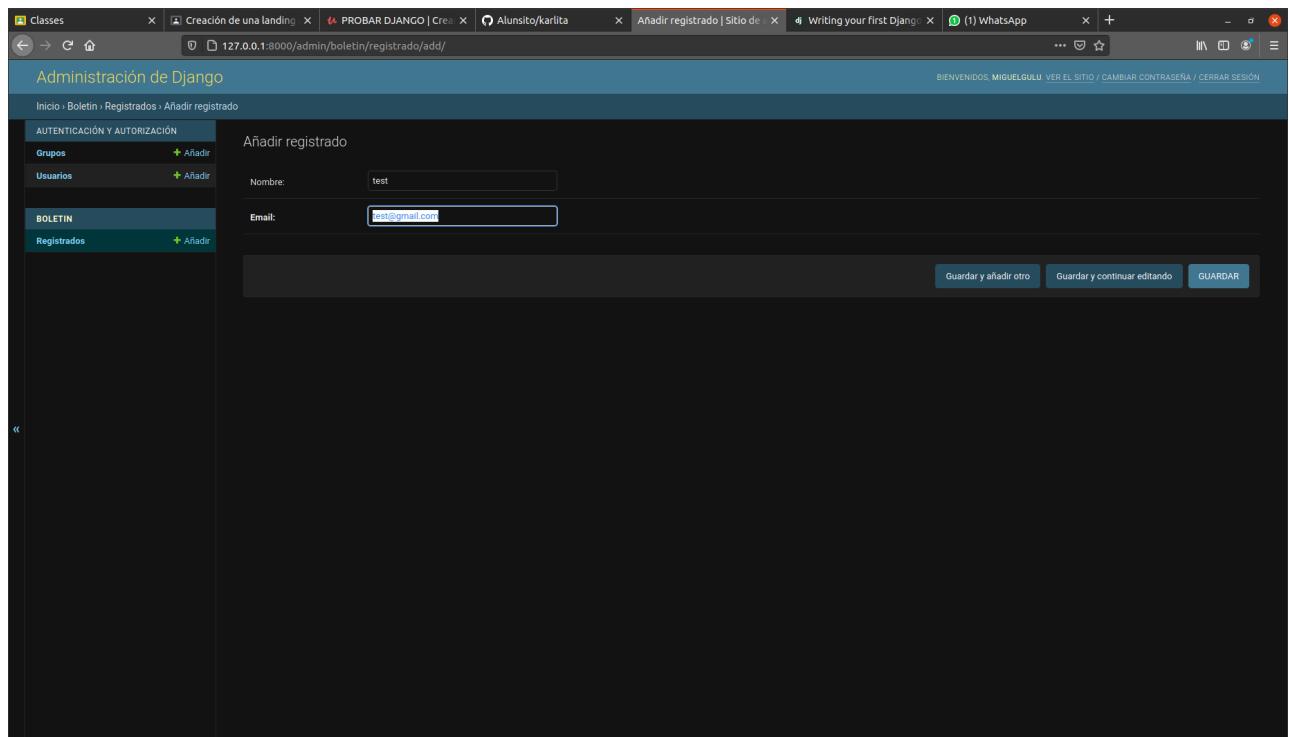


```
1  from django.db import models
2
3  # Create your models here.
4
5  class Registrado(models.Model):
6      nombre = models.CharField(max_length=100, blank=True, null=True)
7      email = models.EmailField()
8      timestamp = models.DateTimeField(auto_now_add=True, auto_now=False)
9
10     def __str__(self):
11         return self.nombre
```

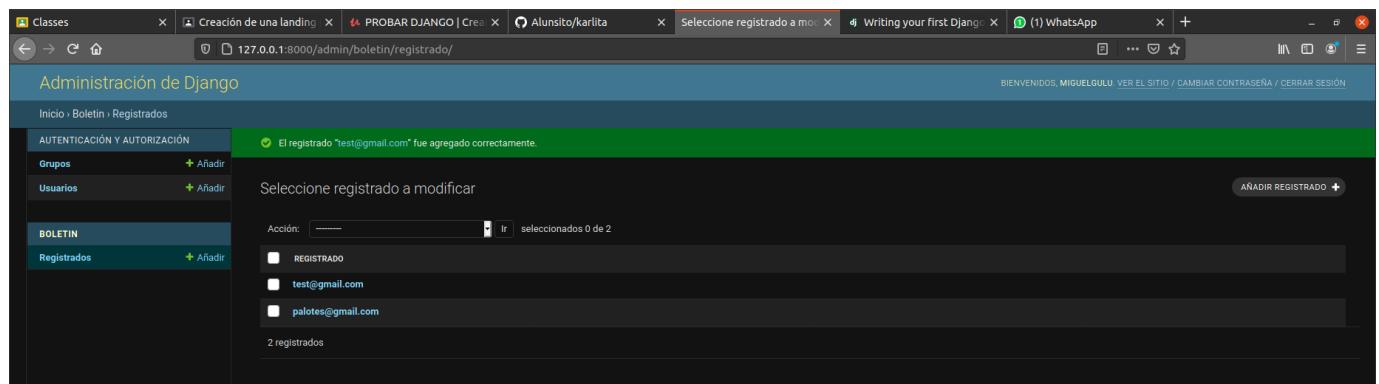
Y si recargamos la página, podemos visualizar los nombres. Una vez comprobado vuelvo a cambiar para que se vea el email.



Para terminar, vamos a añadir un nuevo registro. Para ello, clickamos en “añadir registrado” y completamos los campos:



Y si volvemos otra vez al modelo, podemos ver el nuevo usuario:



8. PERSONALIZAR MODELO EN ADMIN

A continuación, vamos a personalizar nuestro modelo en admin. Para ello nos vamos a nuestro fichero de admin.py y creamos una clase para nuestro modelo.

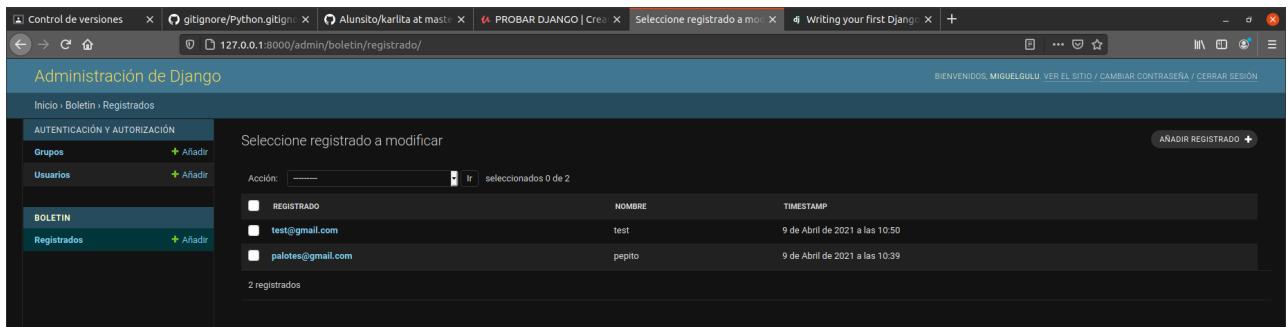


```

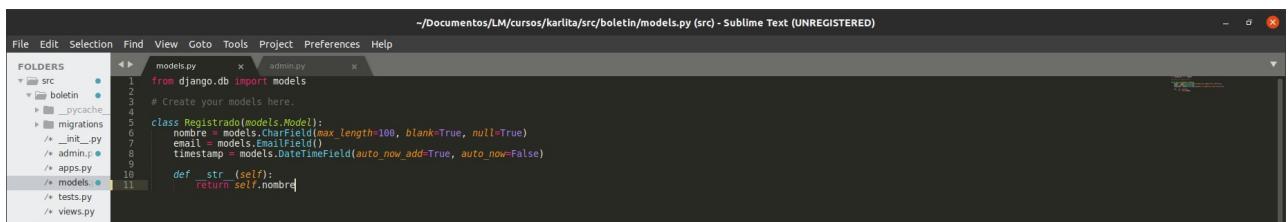
File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS
src
└── boletin
    ├── migrations
    ├── __init__.py
    ├── admin.py
    ├── apps.py
    ├── models.py
    ├── tests.py
    └── views.py
    └── pfolio
        └── __pycache__
            └── admin.py
model.py
admin.py
~ /Documentos/LM/cursos/karlita/src/boletin/admin.py (src) - Sublime Text (UNREGISTERED)
1 from django.contrib import admin
2
3 # Register your models here.
4
5 from models import *
6
7 class AdminRegistrado(admin.ModelAdmin):
8     list_display = ['__str__', 'nombre', 'timestamp']
9     class Meta:
10         model = Registrado
11
12
13 admin.site.register(Registrado, AdminRegistrado)

```

Escribimos “`__str__`” en lugar de “`__unicode__`” porque recuerdo que estamos usando Python 3. En nuestro modelo veremos el nombre, email y fecha de creación.



Si en vez de “`__str__`” escribimos `email` pues saldrá como nombre de la columna “`email`”. Si al modelo le pedimos que devuelva el nombre, en el `admin` se nos mostrará así:



```

File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS
src
└── boletin
    ├── migrations
    ├── __init__.py
    ├── admin.py
    ├── apps.py
    ├── models.py
    ├── tests.py
    └── views.py
models.py
admin.py
~ /Documentos/LM/cursos/karlita/src/boletin/models.py (src) - Sublime Text (UNREGISTERED)
1 from django.db import models
2
3 # Create your models here.
4
5 class Registrado(models.Model):
6     nombre = models.CharField(max_length=100, blank=True, null=True)
7     email = models.EmailField()
8     timestamp = models.DateTimeField(auto_now_add=True, auto_now=False)
9
10    def __str__(self):
11        return self.nombre

```

Administración de Django

Inicio > Boletín > Registrados

Selección registrado a modificar

Acción:	REGISTRADO	NOMBRE	TIMESTAMP
<input type="checkbox"/>	test	test	9 de Abril de 2021 a las 10:50
<input type="checkbox"/>	pepito	pepito	9 de Abril de 2021 a las 10:39

2 registrados

```

File Edit Selection Find View Goto Tools Project Preferences Help
-/Documentos/LM/cursos/karlita/src/boletin/admin.py - Sublime Text (UNREGISTERED)
FOLDERS
src
└── boletin
    ├── __pycache__
    ├── migrations
    ├── __init__.py
    ├── admin.py
    ├── apps.py
    ├── models.py
    ├── tests.py
    └── views.py
pd110
└── __pycache__
    └── __init__.py

models.py
1 from django.contrib import admin
2 # Register your models here.
3
4 from .models import *
5
6 class AdminRegistrado(admin.ModelAdmin):
7     list_display = ['email', 'timestamp']
8     list_filter = ['timestamp']
9     list_editable = ['email']
10    search_fields = ['email', 'nombre']
11
12    class Meta:
13        model = Registrado
14
15
16 admin.site.register(Registrado, AdminRegistrado)

```

Una vez viendo como funciona el display, editaremos los filtros y los campos. Para ello añadiremos estas nuevas líneas de código:

Si dejamos esto tal cual, nos saltará un error debido a que no podemos poner como editable un campo que se utiliza como nombre de columna y que no es editable. Lo corregimos y quedará algo tal que así:

```

File Edit Selection Find View Goto Tools Project Preferences Help
-/Documentos/LM/cursos/karlita/src/boletin/admin.py - Sublime Text (UNREGISTERED)
FOLDERS
src
└── boletin
    ├── __pycache__
    ├── migrations
    ├── __init__.py
    ├── admin.py
    ├── apps.py
    ├── models.py
    ├── tests.py
    └── views.py
pd110
└── __pycache__
    └── __init__.py

models.py
1 from django.contrib import admin
2 # Register your models here.
3
4 from .models import *
5
6 class AdminRegistrado(admin.ModelAdmin):
7     list_display = ['str', 'email', 'timestamp']
8     list_filter = ['timestamp']
9     list_editable = ['email']
10    search_fields = ['email', 'nombre']
11
12    class Meta:
13        model = Registrado
14
15
16 admin.site.register(Registrado, AdminRegistrado)

```

EMAIL	NOMBRE	TIMESTAMP
test@gmail.com	test	9 de Abril de 2021 a las 10:50
palotes@gmail.com	pepito	9 de Abril de 2021 a las 10:39

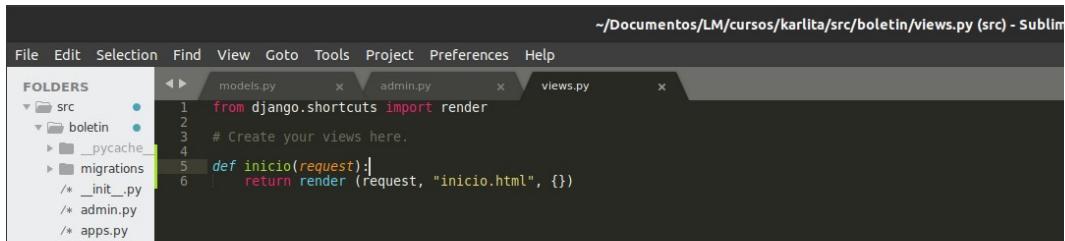
Y si volvemos a nuestro *admin site* podemos ver que podemos editar el campo de nombre sin necesidad de modificarlo entrando directamente a cada registro uno a uno. Podemos ver que tenemos también los filtros que hemos puesto en la lista de filtros.

Si quisiéramos solucionar el error anterior pero dejando que aparezca el nombre en la columna, hay que añadir al código una línea de **list_display_links** en el cual añadimos el nombre. Una vez hecho saldrá como queríamos:

The screenshot shows the Django Admin interface for managing users. The top navigation bar includes links for 'Control de versiones', 'gitignore/Python.gitignore', 'Alunlito/karlita at master', 'PROBAR DJANGO | Crear', 'Selección de usuario a modificar', 'Writing your first Django app', and '+'. The main title is 'Administración de Django' with a subtitle 'Inicio > Autenticación y autorización > Usuarios'. On the left, a sidebar lists 'AUTENTICACIÓN Y AUTORIZACIÓN' with 'Grupos' and 'Usuarios' (selected), and 'BOLETIN' with 'Registrados' (selected). The main content area is titled 'Seleccione usuario a modificar' and contains a search bar and a table with two rows. The table columns are 'NOMBRE DE USUARIO' (checkboxes for 'jcanillas' and 'miguelgulu'), 'DIRECCIÓN DE CORREO ELECTRÓNICO' (values 'jcanillas' and 'miguel.guerrero.luna.alu@iesfernandoaguilar.es'), 'NOMBRE', 'APELIDOS', and 'ES STAFF' (radio buttons for 'jcanillas' and 'miguelgulu'). A 'FILTRO' sidebar on the right allows filtering by 'Por es staff' (Todo, Sí, No), 'Por estado de superusuario' (Todo, Sí, No), and 'Por activo' (Todo, Sí, No). A 'BUSCAR' button is also present.

9. PRIMERA VISTA

Si queremos crear nuestra primera vista de nuestra aplicación debemos abrir, como era de esperar, el archivo de “views.py”. Hay que definir una función que devuelva una plantilla de la vista que vamos a crear. Nuestra plantilla será “*inicio.html*”.



```

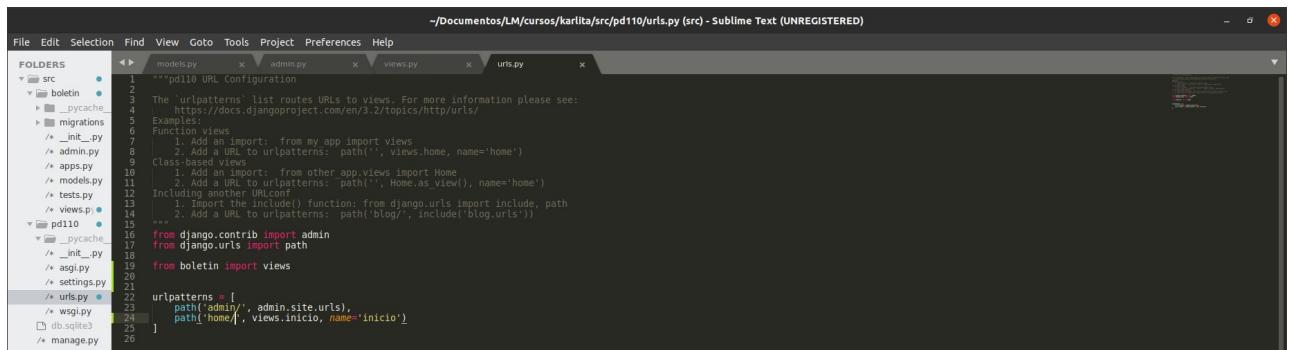
File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS models.py admin.py views.py
src
└── boletin
    ├── __pycache__
    ├── migrations
    ├── __init__.py
    ├── admin.py
    └── apps.py

1 from django.shortcuts import render
2
3 # Create your views here.
4
5 def inicio(request):
6     return render (request, "inicio.html", {})

~/Documentos/LM/cursos/karlita/src/boletin/views.py (src) - Sublime Text

```

Una vez tengamos nuestra vista establecida, tenemos que crear la url que corresponda a nuestra vista. Abrimos el archivo “*urls.py*” y escribimos una nueva línea donde se definen los parámetros de nuestra url. Como Karlita usa Python 2, utiliza otra estructura y la función ‘url’, pero como estoy usando Python 3 necesito la función ‘path’ (todo esto está en lo comentado que aparece arriba).



```

File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS models.py admin.py views.py urls.py
src
└── boletin
    ├── __pycache__
    ├── migrations
    ├── __init__.py
    ├── admin.py
    ├── apps.py
    ├── models.py
    ├── tests.py
    └── views.py

1 """pd110 URL Configuration
2
3 The 'urlpatterns' list routes URLs to views. For more information please see:
4     https://docs.djangoproject.com/en/3.2/topics/http/urls/
5 Examples:
6     Function views
7         1. Add an import: from my_app import views
8             2. Add a URL to urlpatterns: path('', views.home, name='home')
9 Class-based views
10    1. Add an import: from other_app.views import Home
11    2. Add a URL to urlpatterns: path('', Home.as_view(), name='home')
12 Including another URLconf
13     1. Import the include() function: from django.urls import include, path
14     2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
15 ...
16
17 from django.contrib import admin
18 from django.urls import path
19
20 from boletin import views
21
22 urlpatterns = [
23     path('admin/', admin.site.urls),
24     path('home/', views.inicio, name='inicio')
25 ]
26

~/Documentos/LM/cursos/karlita/src/pd110/urls.py (src) - Sublime Text (UNREGISTERED)

```

Yo he escrito para nuestra vista la url de ‘**127.0.0.1:8000/home**’, pero luego decidí usar la url ‘**127.0.0.1:8000**’ a secas, ya que será nuestra página de inicio. Si guardamos y abrimos la url saldrá algo tal que así:

TemplateDoesNotExist at /
inicio.html

Request Method: GET
Request URL: http://127.0.0.1:8000/
Django Version: 3.2
Exception Type: TemplateDoesNotExist
Exception Value: inicio.html
Exception Location: /home/alunsito/Documentos/LM/cursos/karita/venv/lib/python3.8/site-packages/django/template/loader.py, line 19, in get_template
Python Executable: /home/alunsito/Documentos/LM/cursos/karita/venv/bin/python3
Python Version: 3.8.5
Python Path: ['/home/alunsito/Documentos/LM/cursos/karita/src', '/usr/lib/python3.8', '/usr/lib/python3.8/lib-dynload', '/home/alunsito/Documentos/LM/cursos/karita/venv/lib/python3.8/site-packages']
Server time: Fri, 09 Apr 2021 19:04:20 +0000

Template-loader postmortem

Django tried loading these templates, in this order:

Using engine django:
• django.template.loaders.app_directories.Loader: /home/alunsito/Documentos/LM/cursos/karita/venv/lib/python3.8/site-packages/django/contrib/admin/templates/inicio.html (Source does not exist)
• django.template.loaders.app_directories.Loader: /home/alunsito/Documentos/LM/cursos/karita/venv/lib/python3.8/site-packages/django/contrib/auth/templates/inicio.html (Source does not exist)

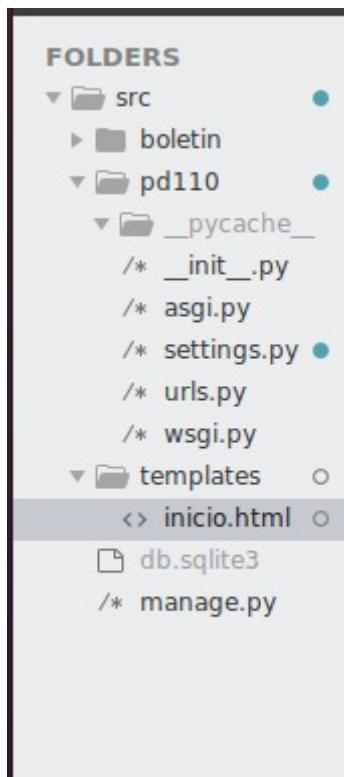
Traceback Switch to copy-and-paste view

```
/home/alunsito/Documentos/LM/cursos/karita/venv/lib/python3.8/site-packages/django/core/handlers/exception.py, line 47, in inner
    47.     response = get_response(request)
▶ Local vars
/home/alunsito/Documentos/LM/cursos/karita/venv/lib/python3.8/site-packages/django/core/handlers/base.py, line 181, in _get_response
    181.     response = wrapped_callback(request, *callback_args, **callback_kwargs)
▶ Local vars
/home/alunsito/Documentos/LM/cursos/karita/src/boletin/views.py, line 6, in inicio
    6.     return render (request, "inicio.html", {})
▶ Local vars
/home/alunsito/Documentos/LM/cursos/karita/venv/lib/python3.8/site-packages/django/shortcuts.py, line 19, in render
    19.     content = loader.render_to_string(template_name, context, request, using=using)
▶ Local vars
/home/alunsito/Documentos/LM/cursos/karita/venv/lib/python3.8/site-packages/django/template/loader.py, line 61, in render_to_string
    61.     template = get_template(template_name, using=using)
▶ Local vars
```

Nos salta un error el cual nos dice que no existe la plantilla. Eso se debe a que “*inicio.html*” no está creado y por lo cual no se puede mostrar la vista. Esto lo solucionaremos en el siguiente capítulo.

10. CONFIGURACIÓN DE PLANTILLAS

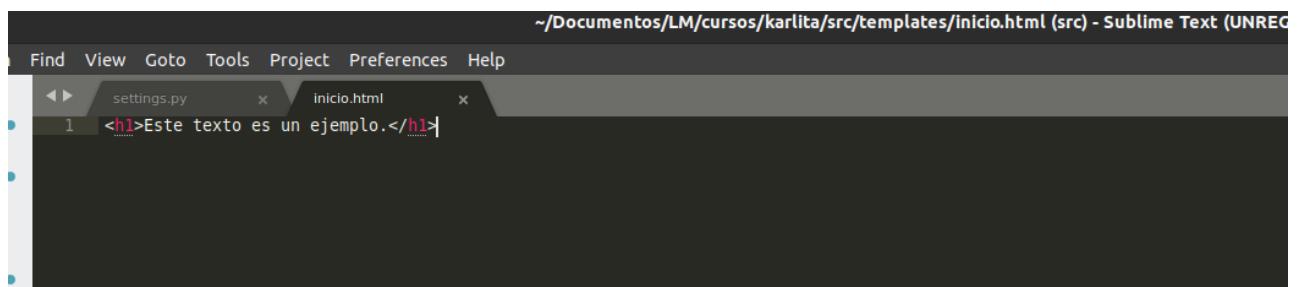
Llegó el momento de crear nuestra página de inicio de nuestra aplicación. Para ello, debemos crear la carpeta de ‘*templates*’ con el archivo de ‘*inicio.html*’ en su interior:



Una vez creado todo debemos añadir la carpeta al proyecto para que, en caso de querer usar todos los recursos que almacene, los tengamos de manera accesible. La manera de hacer esto es en archivo ‘*settings.py*’ y, dentro de **TEMPLATES**, añadir la dirección del directorio. La estructura para Python 3 es como se indica en la siguiente captura, en Python 2 sería de otra forma (en los comentarios del archivo viene como sería).

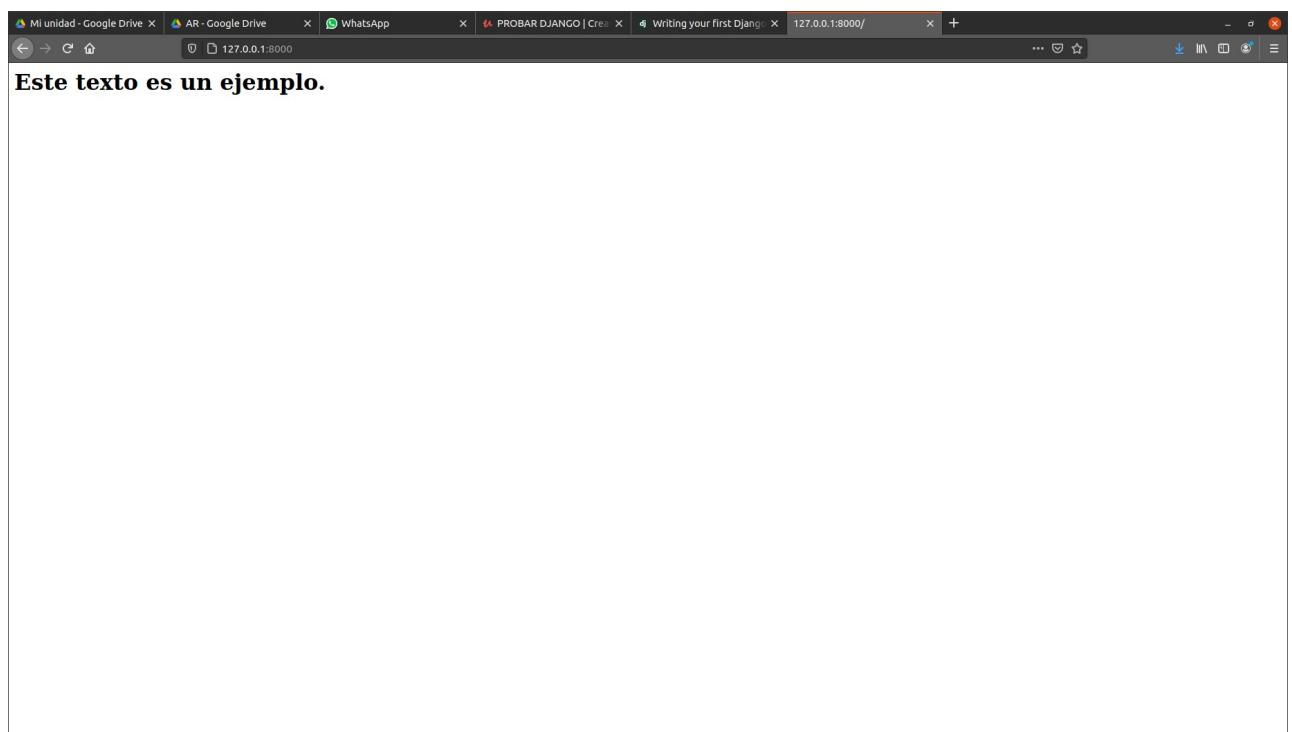
```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [BASE_DIR / 'templates'],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]
```

Una vez hecho esto, creamos una plantilla minimalista solamente para hacer la prueba:



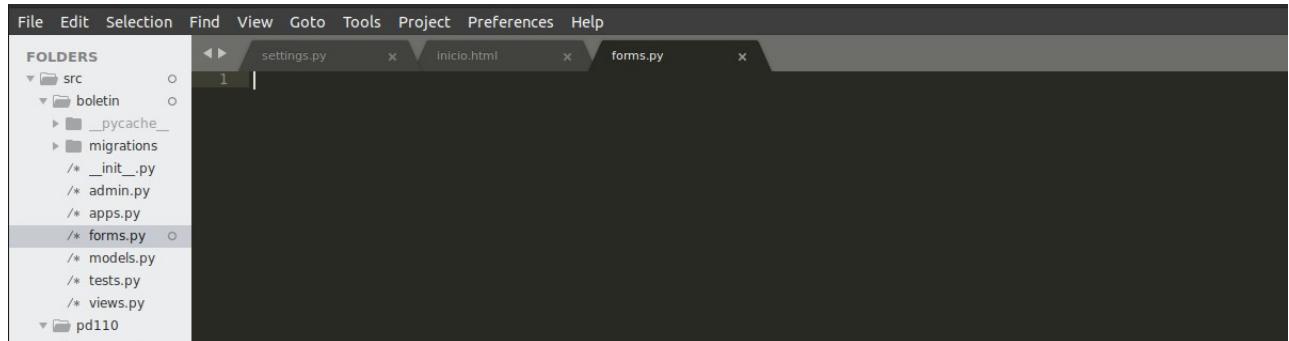
```
~/Documentos/LM/cursos/karlita/src/templates/inicio.html (src) - Sublime Text (UNRECORDED)
Find View Goto Tools Project Preferences Help
settings.py      inicio.html
1  <h1>Este texto es un ejemplo.</h1>
```

Y, habiéndolo guardado, abrimos la dirección donde tenemos nuestra vista en el navegador:



11. ESCRIBIR FORMULARIO

Para crear nuestro primer formulario debemos crear el archivo que lo contiene, en este caso será ‘forms.py’:



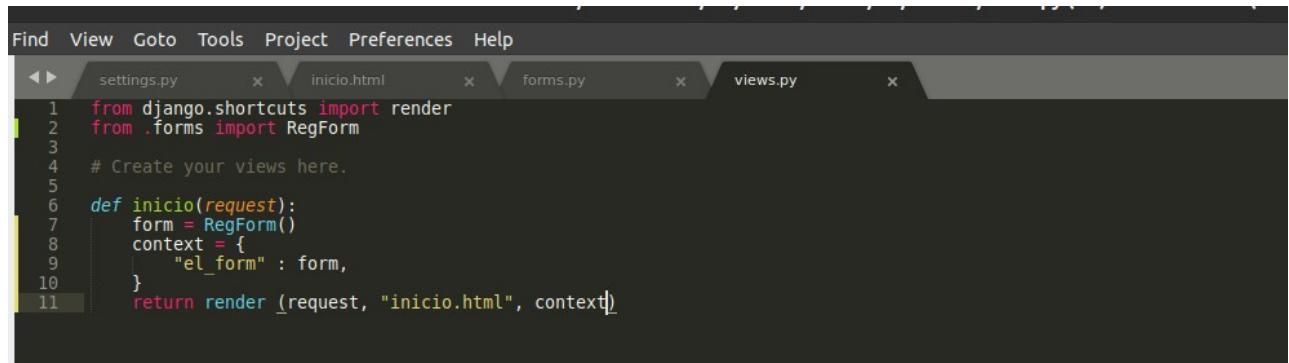
Vamos a crear un formulario básico, el cual se rellena con el nombre y la edad:

```
1 from django import forms
2
3 class RegForm(forms.Form):
4     nombre = forms.CharField(max_length=100)
5     edad = forms.IntegerField()
```

Estos son campos que habrán que llenar cuando se vayan a crear nuevos usuarios. Como veremos en siguientes capítulo, hay distintas maneras de hacerlo.

12. FORMULARIO EN UNA VISTA

Para aplicar nuestro formulario en nuestra vista, nos dirigimos a nuestro ‘views.py’ e importamos nuestra función del formulario.

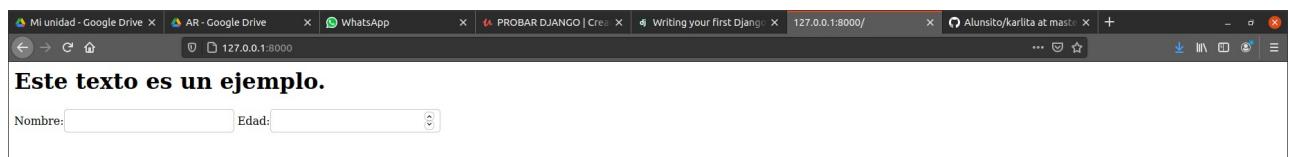


```

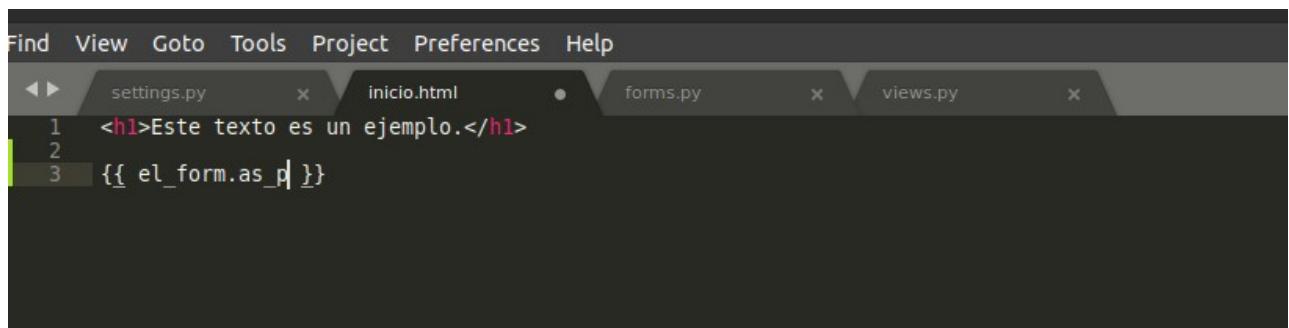
Find View Goto Tools Project Preferences Help
  settings.py      inicio.html      forms.py      views.py
1 from django.shortcuts import render
2 from .forms import RegForm
3
4 # Create your views here.
5
6 def inicio(request):
7     form = RegForm()
8     context = {
9         "el_form": form,
10    }
11    return render (request, "inicio.html", context)

```

Y dentro de la función, definimos como variable la función del formulario que creamos. En ‘context’ están los campos y en ‘return’ devuelve el request, la plantilla y los campos a llenar.



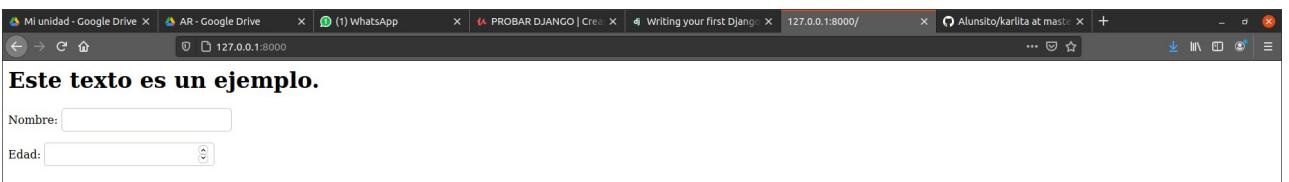
Vamos a hacerle un cambio para que se vea mejor poniendo ambos campos cada uno en una línea.



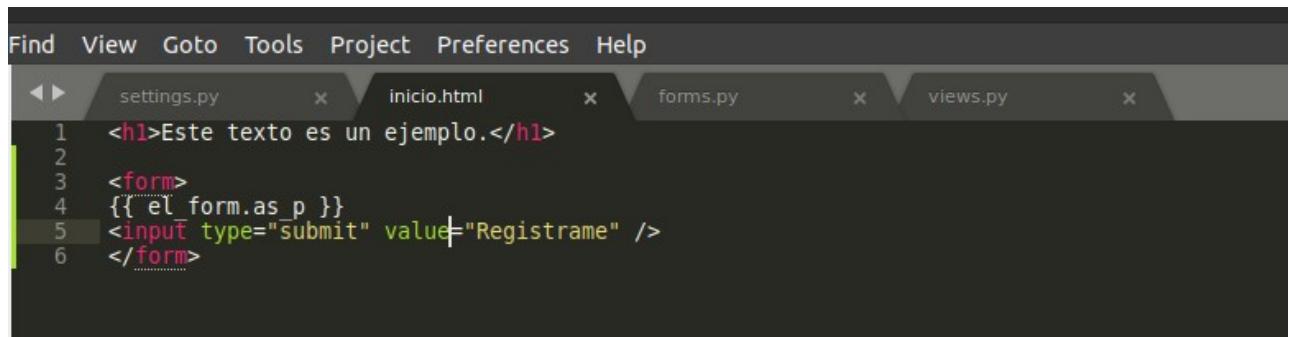
```

Find View Goto Tools Project Preferences Help
  settings.py      inicio.html      forms.py      views.py
1 <h1>Este texto es un ejemplo.</h1>
2
3 {{ el_form.as_p }}

```



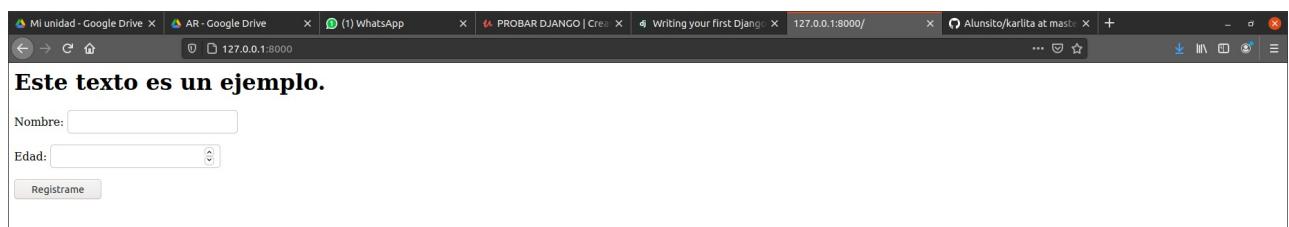
Para terminar, vamos a crear el botón que nos dará la opción de registrarnos en el formulario. Para ello añadimos un *input* tipo *submit*:



A screenshot of a code editor showing a Python template file named 'inicio.html'. The code contains the following HTML:

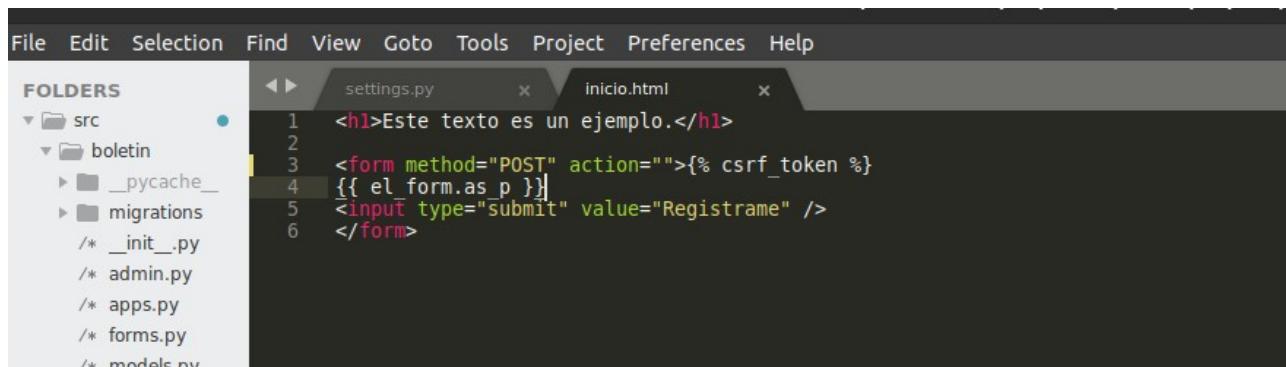
```
1 <h1>Este texto es un ejemplo.</h1>
2
3 <form>
4 {{ el_form.as_p }}
5 <input type="submit" value="Regístrate" />
6 </form>
```

Entonces en la vista saldrá tal que así:



13. MÉTODO HTTP POST EN FORMULARIO

En este capítulo vamos a trabajar con el método HTTP POST en nuestro formulario. Si queremos añadirlo, en la etiqueta *form* debemos añadir el ‘método POST’. Tendremos que escribirlo de la siguiente manera...



```
File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS
src
  boletin
    __pycache__
    migrations
    __init__.py
    admin.py
    apps.py
    forms.py
    models.py
settings.py      inicio.html
1 <h1>Este texto es un ejemplo.</h1>
2
3 <form method="POST" action="">{{ csrf_token }}{{ el_form.as_p }}</form>
4
5 <input type="submit" value="Regístrate" />
6
```

Entonces cuando escribamos algo en el formulario y nos registremos como esto:



Este texto es un ejemplo.

Nombre: test

Edad: 20

Regístrate

En nuestro terminal donde ejecutamos la aplicación sale lo siguiente:



Este texto es un ejemplo.

Nombre:

Edad:

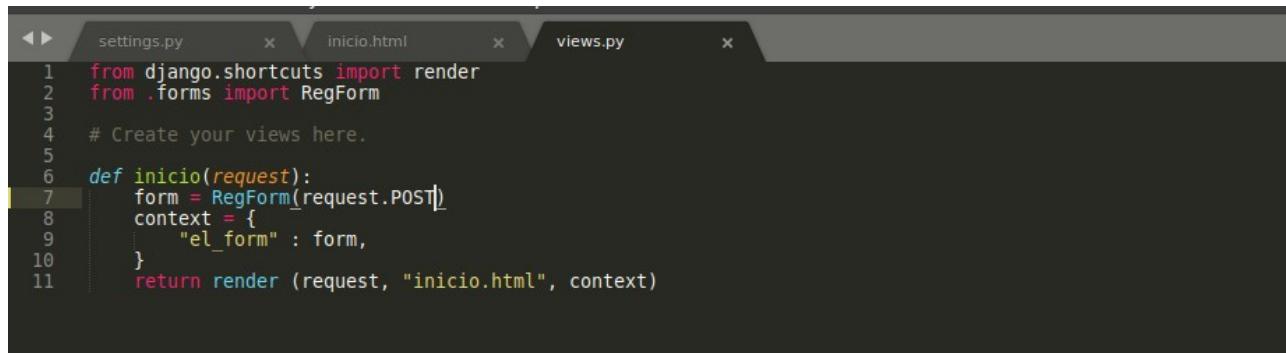
Regístrate

```
System check identified no issues (0 silenced).
April 11, 2021 - 09:58:46
Django version 3.2, using settings 'pd110.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
[11/Apr/2021 09:58:52] "GET / HTTP/1.1" 200 456
Not Found: /favicon.ico
[11/Apr/2021 10:00:16] "POST / HTTP/1.1" 200 456
```

Al final se ve el resto del método HTTP POST que pretendíamos establecer.

14. VALIDACIONES FORMULARIOS PT. 1

En esta parte del curso nos vamos a enfocar un poco en como validar los formularios que rellenemos en nuestra vista. Para asegurarnos de que todos los datos entran de una manera limpia tendremos que irnos a nuestra vista y escribimos ‘request.POST’ aquí:



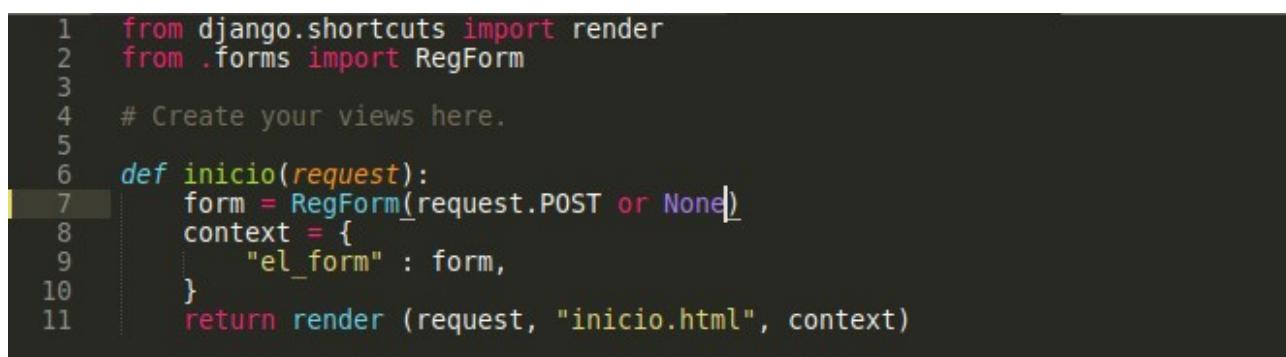
```
1 from django.shortcuts import render
2 from .forms import RegForm
3
4 # Create your views here.
5
6 def inicio(request):
7     form = RegForm(request.POST)
8     context = {
9         "el_form": form,
10    }
11    return render (request, "inicio.html", context)
```

Entonces cuando recargamos nuestra vista en el navegador sale algo tal que así:



Aquí nos exige, con las casillas bordeadas en rojo, que los campos son obligatorios. Si no se llenan todos no se declarará el formulario como completado.

Si añadimos ‘or None’, solamente se nos marcará que hay que llenarlo obligatoriamente en el caso de que le demos a registrarnos:



```
1 from django.shortcuts import render
2 from .forms import RegForm
3
4 # Create your views here.
5
6 def inicio(request):
7     form = RegForm(request.POST or None)
8     context = {
9         "el_form": form,
10    }
11    return render (request, "inicio.html", context)
```



Si escribimos esta nueva línea en nuestro código, cuando vamos a nuestro terminal donde hemos arrancado nuestra aplicación, podemos ver todos las opciones relacionadas con los formularios que podemos considerar:

```
$ print(dir(form))
```

```
alunsilto@alunsilto: ~/Documentos/LM/cursos/karlita/src
Performing system checks...
System check identified no issues (0 silenced).
April 11, 2021 - 10:18:54
Django version 3.2, using settings 'pd10.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
/home/alunsilto/Documentos/LM/cursos/karlita/src/boletin/views.py changed, reloading.
Watching for file changes with StatReloader
Performing system checks...
System check identified no issues (0 silenced).
April 11, 2021 - 10:19:03
Django version 3.2, using settings 'pd10.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
['__class__', '__delattr__', '__dict__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__gt__', '__hash__', '__html__', '__init__', '__init_subclass__', '__iter__', '__le__', '__lt__', '__module__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__', '__weakref__', '__bound_fields_cache__', '__clean_fields__', '__clean_form__', '__errors__', '__html_output__', '__post_clean__', '__add_error__', '__add_initial_prefix__', '__add_prefix__', '__as_p__', '__as_table__', '__as_ul__', '__auto_id__', '__base_fields__', '__changed_data__', '__clean__', '__data__', '__declared_fields__', '__default_renderer__', '__empty_permitted__', '__error_class__', '__errors__', '__field_order__', '__fields__', '__files__', '__full_clean__', '__get_initial_for_field__', '__has_changed__', '__has_error__', '__hidden_fields__', '__initial__', '__is_bound__', '__is_multipart__', '__is_valid__', '__label_suffix__', '__media__', '__non_field_errors__', '__order_fields__', '__prefix__', '__renderer__', '__use_required_attribute__', '__visible_fields__']
[11/Apr/2021 10:19:30] "POST / HTTP/1.1" 200 480
```

Para poder registrar y almacenar todo lo que escribamos en el formulario, tendremos que añadir una condición que consiste en que si el formulario es válido, se muestra por pantalla los datos que hemos introducido:

```
1 from django.shortcuts import render
2 from .forms import RegForm
3
4 # Create your views here.
5
6 def inicio(request):
7     form = RegForm(request.POST or None)
8     if form.is_valid():
9         print (form.cleaned_data)
10    context = {
11        "el_form" : form,
12    }
13    return render (request, "inicio.html", context)
```

```
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
{'nombre': 'test', 'edad': 20}
[11/Apr/2021 10:21:25] "POST / HTTP/1.1" 200 480
```

Como podemos ver, por pantalla sale un diccionario el cual contiene como claves los campos y como valor lo que hemos escrito.

Este formato de salida podemos modificarlo a nuestro antojo, como por ejemplo declarando este print como variable de diccionario y decidiendo que parte de la misma mostrar. En este caso, vamos a hacer que muestre nombre y edad en líneas separadas:

```
1 from django.shortcuts import render
2 from .forms import RegForm
3
4 # Create your views here.
5
6 def inicio(request):
7     form = RegForm(request.POST or None)
8     if form.is_valid():
9         form_data = form.cleaned_data
10        print (form_data.get("nombre"))
11        print (form_data.get("edad"))
12    context = {}
13    'el_form' : form,
14
15 return render (request, "inicio.html", context)
```

Y con este código sale algo tal que así:

```
test
20
[11/Apr/2021 10:23:08] "POST / HTTP/1.1" 200 480
test
20
[11/Apr/2021 10:23:14] "POST / HTTP/1.1" 200 480
test
20
[11/Apr/2021 10:23:18] "POST / HTTP/1.1" 200 480
```

15. GUARDAR DATOS DEL FORMULARIO CON EL MODELO

Para guardar los datos vamos, en primer lugar, vamos a cambiar los campos del formulario para que sea de acorde al modelo que tenemos, el cual tiene ‘nombre’, ‘email’ y ‘timestamp’. Vamos a añadir solo ‘nombre’ e ‘email’:

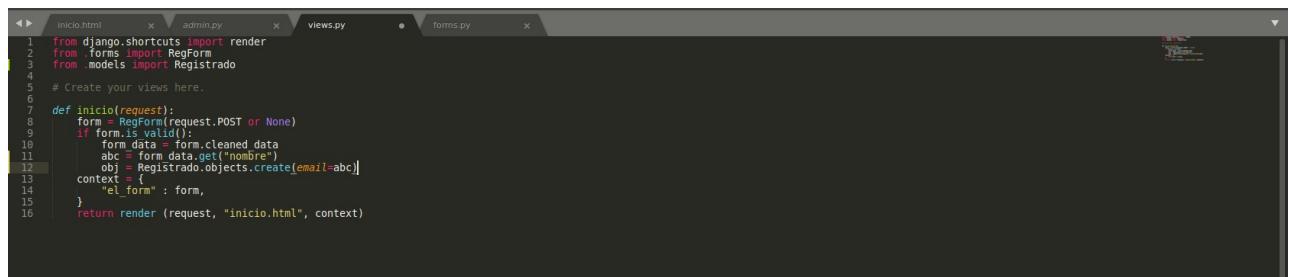


```

1 from django import forms
2
3 class RegForm(forms.Form):
4     nombre = forms.CharField(max_length=100)
5     email = forms.EmailField()

```

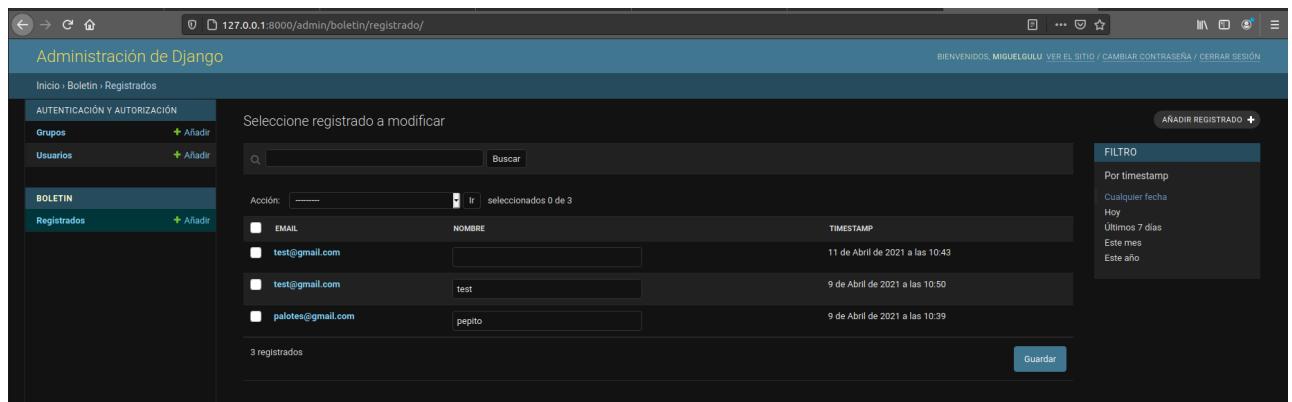
Nos vamos a nuestra vista y la configuraremos para que nos pida los campos que hemos mencionado en el modelo y que almacene en ‘email’ lo que pongamos en el campo de email (me confundí y escribí nombre en vez de email, entonces se pondrá nombre en el email. Para solucionarlo habría que cambiar ese nombre por email):



```

1 from django.shortcuts import render
2 from .forms import RegForm
3 from .models import Registrado
4
5 # Create your view here.
6
7 def inicio(request):
8     form = RegForm(request.POST or None)
9     if form.is_valid():
10         data = form.cleaned_data
11         abc = form.data.get("nombre")
12         obj = Registrado.objects.create(email=abc)
13     context = {
14         "el_form": form,
15     }
16     return render (request, "inicio.html", context)

```

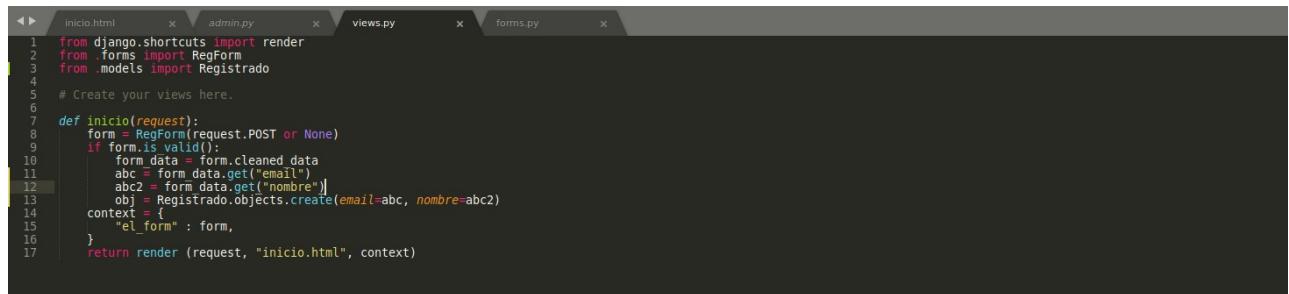


The screenshot shows the Django Admin interface at the URL `127.0.0.1:8000/admin/boletin/registrado/`. The left sidebar has 'AUTENTICACIÓN Y AUTORIZACIÓN' and 'BOLETIN' sections, with 'Registrados' selected. The main area title is 'Seleccione registrado a modificar'. It lists three registered users with their email, name, and timestamp. A 'FILTRO' sidebar on the right shows timestamp filters like 'Por timestamp', 'Hoy', 'Últimos 7 días', 'Este mes', and 'Este año'. At the bottom right is a 'Guardar' button.

EMAIL	NOMBRE	TIMESTAMP
test@gmail.com		11 de Abril de 2021 a las 10:43
test@gmail.com	test	9 de Abril de 2021 a las 10:50
palotes@gmail.com	pepito	9 de Abril de 2021 a las 10:39

Y como podéis ver, en la columna de email en nuestro admin está el correo que hemos escrito en el formulario pero sin nombre, ya que solamente hemos escrito para que almacene el email.

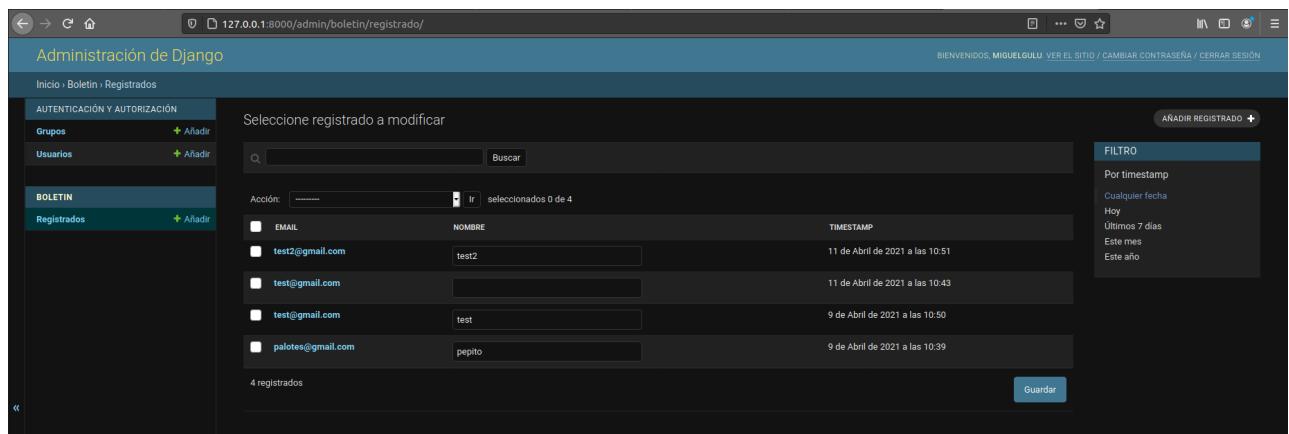
Ahora vamos a modificar el código para que almacene las dos cosas:



```

1  from django.shortcuts import render
2  from .forms import RegForm
3  from .models import Registrado
4
5  # Create your views here.
6
7  def inicio(request):
8      form = RegForm(request.POST or None)
9      if form.is_valid():
10          form_data = form.cleaned_data
11          abc = form_data.get("email")
12          abc2 = form_data.get("nombre")
13          obj = Registrado.objects.create(email=abc, nombre=abc2)
14          context = {
15              "el_form": form,
16          }
17      return render (request, "inicio.html", context)

```

EMAIL	NOMBRE	TIMESTAMP
test2@gmail.com	test2	11 de Abril de 2021 a las 10:51
test@gmail.com		11 de Abril de 2021 a las 10:43
test@gmail.com	test	9 de Abril de 2021 a las 10:50
palotes@gmail.com	pepito	9 de Abril de 2021 a las 10:39

Y finalmente, los campos que hemos rellenado en nuestro formulario tanto nombre como email los tenemos almacenados en nuestra base.

16. MODEL FORM

Para crear un modelo más dinámico en el cual registrar nuestros datos vamos a crear nuestro model form. Para ello, nos vamos a nuestro ‘forms.py’ e importamos nuestro modelo.

A continuación creamos una nueva clase y pondremos como campo a llenar (el cual se rellena en *admin*) el cual solo pida el *email*:

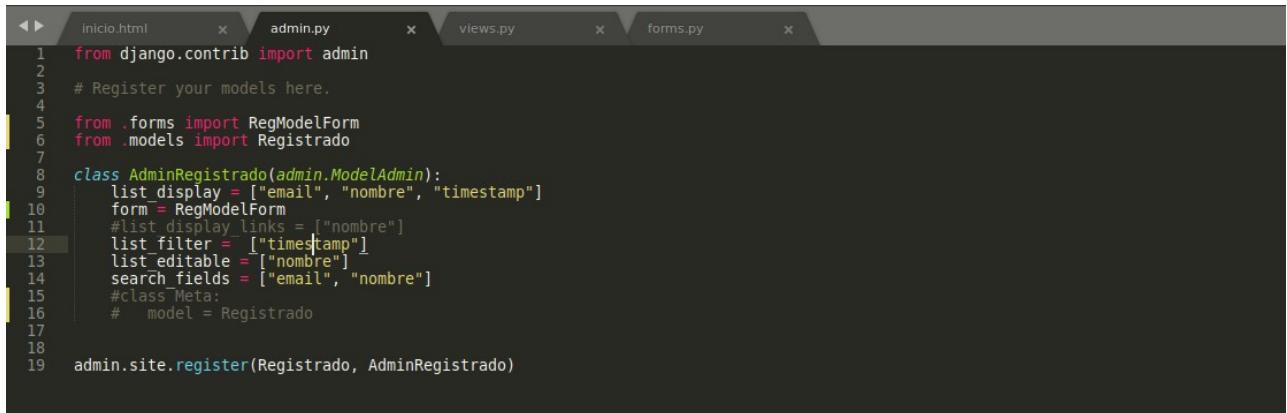


```

1  from django import forms
2
3  from .models import Registrado
4
5  class RegModelForm(forms.ModelForm):
6      class Meta:
7          model = Registrado
8          fields = ["email"]
9
10 class RegForm(forms.Form):
11     nombre = forms.CharField(max_length=100)
12     email = forms.EmailField()

```

Y en nuestro ‘*admin.py*’ importamos nuestro modelo y añadimos como formulario la clase que creamos anteriormente:

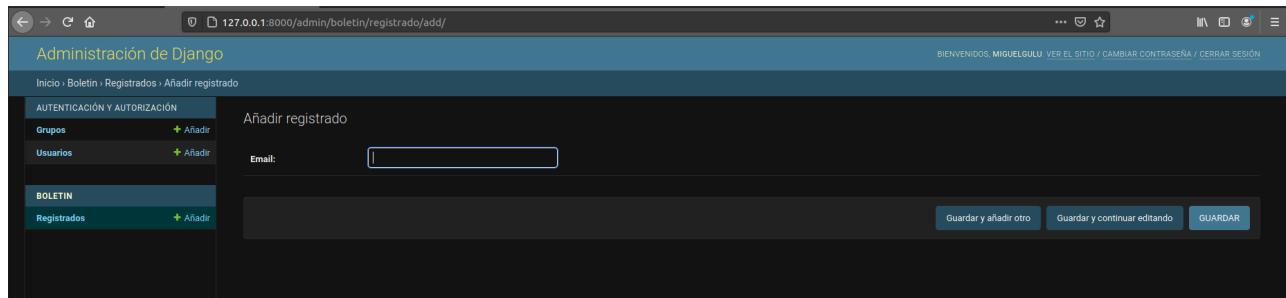


```

1  from django.contrib import admin
2
3  # Register your models here.
4
5  from .forms import RegModelForm
6  from .models import Registrado
7
8  class AdminRegistrado(admin.ModelAdmin):
9      list_display = ["email", "nombre", "timestamp"]
10     form = RegModelForm
11     #list_display_links = ["nombre"]
12     list_filter = ["timestamp"]
13     list_editable = ["nombre"]
14     search_fields = ["email", "nombre"]
15     #class Meta:
16     #    model = Registrado
17
18 admin.site.register(Registrado, AdminRegistrado)

```

Y en nuestro *admin* saldrá esto:

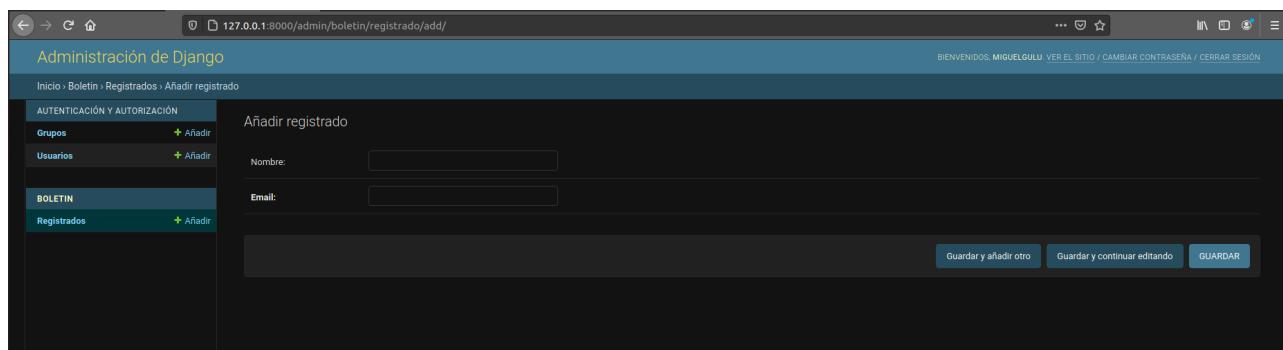


Solamente con llenar el campo de *email* ya podríamos añadir un nuevo registro.

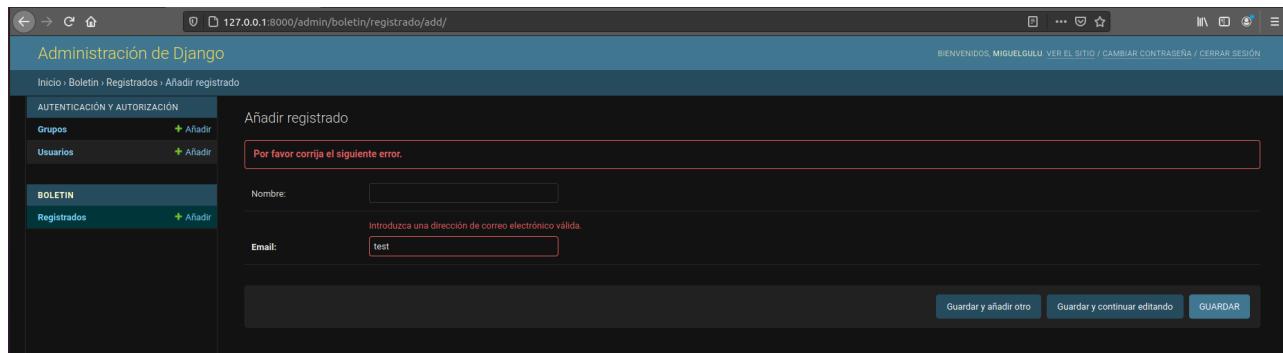
Vamos a ajustarlo a como lo teníamos al principio pero seguimos usando nuestro *Model Form*:

```
1  from django import forms
2
3  from .models import Registrado
4
5  class RegModelForm(forms.ModelForm):
6      class Meta:
7          model = Registrado
8          fields = ["nombre", "email"]
9
10 class RegForm(forms.Form):
11     nombre = forms.CharField(max_length=100)
12     email = forms.EmailField()
```

Ahora tenemos los campos que teníamos al principio.



Si introducimos un *email* que no tiene la estructura de uno...



Nos dará error, ya que al ser un *EmailField* tiene que llevar un '@' y un '.'.

17. VALIDACIONES MODEL FORM

Como vimos en la práctica anterior, al final, *django* tiene sus propias maneras de validar los campos que rellenamos en el modelo. En esta ocasión vamos a ver como añadir nuestras propias validaciones. Primero, debemos abrir nuestro *Model Form* en nuestro ‘*forms.py*’ y añadir una nueva función dentro de la clase la cual creará un email predeterminado para todos los registros que hagamos. El código quedaría de esta manera:

```

1  from django import forms
2
3  from .models import Registrado
4
5  class RegModelForm(forms.ModelForm):
6      class Meta:
7          model = Registrado
8          fields = ["nombre", "email"]
9
10     def clean_email(self):
11         print(self.cleaned_data)
12         return "email@email.com"
13
14 class RegForm(forms.Form):
15     nombre = forms.CharField(max_length=100)
16     email = forms.EmailField()

```

Entonces cada vez que hagamos un registro, el *email* será ‘*email@email.com*’:

The screenshot shows the Django Admin 'Registrado' list view. At the top, a green success message says 'El registrado "email@email.com" fue agregado correctamente.' On the right, there's a 'FILTRO' sidebar with options like 'Por timestamp', 'Cualquier fecha', 'Hoy', 'Últimos 7 días', 'Este mes', and 'Este año'. The main table lists one user: 'EMAIL' (checkbox selected) and 'email@email.com'. The 'NOMBRE' column is empty and the 'TIMESTAMP' column shows '13 de Abril de 2021 a las 08:52'.

Si vamos a nuestro terminal, vemos como se nos devuelve por pantalla el *email* que hemos introducido en el registro:

```

[13/Apr/2021 08:51:28] "GET /admin/boletin/registrado/add/ HTTP/1.1" 200 6209
[13/Apr/2021 08:51:28] "GET /admin/jsi18n/ HTTP/1.1" 200 7640
{'nombre': None, 'email': 'validacion@mail.com'}
[13/Apr/2021 08:52:25] "POST /admin/boletin/registrado/add/ HTTP/1.1" 302 0

```

Después de ver como funciona, podemos añadir nuestras propias validaciones para el modelo. En este caso, vamos poner como requisito que la extensión del email debe ser ‘.edu’. Hay dos maneras de hacerlo. La primera manera es esta:

```

1  from django import forms
2
3  from .models import Registrado
4
5  class RegModelForm(forms.ModelForm):
6      class Meta:
7          model = Registrado
8          fields = ["nombre", "email"]
9
10     def clean_email(self):
11         email = self.cleaned_data.get("email")
12         if not "edu" in email:
13             raise forms.ValidationError("Por favor use un email válido con extensión .edu")
14         return email
15
16 class RegForm(forms.Form):
17     nombre = forms.CharField(max_length=100)
18     email = forms.EmailField()

```

Añadir registrado

Por favor corrija el siguiente error.

Nombre:

Por favor use un email valido con extension .edu

Email:

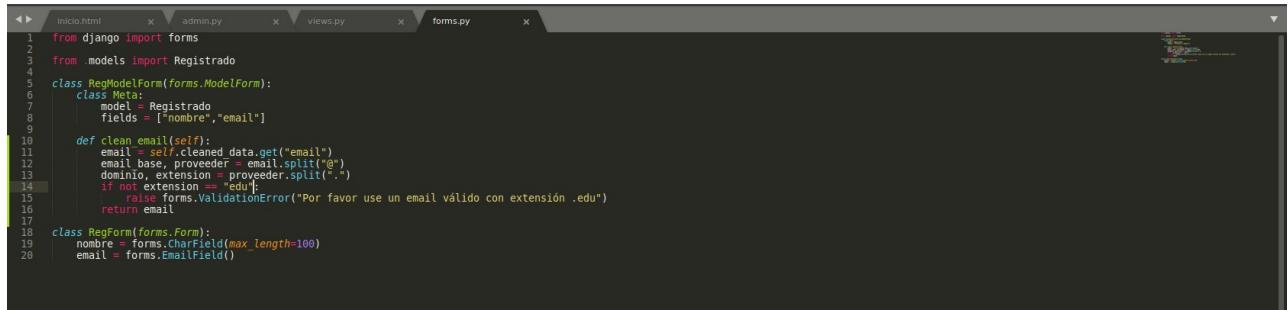
Guardar y añadir otro Guardar y continuar editando GUARDAR

Este método funciona pero es poco eficiente, ya que el requisito es, según hemos puesto en el código, que el email tenga la cadena ‘edu’, entonces si ponemos ‘edu’ en cualquier parte y de cualquier manera en el email nos lo dará por válido:

EMAIL	NOMBRE	TIMESTAMP
validacion@eduardo.com	validacion	13 de Abril de 2021 a las 08:57
email@email.com		13 de Abril de 2021 a las 08:52
test2@gmail.com	test2	11 de Abril de 2021 a las 10:51
test@gmail.com	test	11 de Abril de 2021 a las 10:43
test@gmail.com	pepito	9 de Abril de 2021 a las 10:50
palotes@gmail.com		9 de Abril de 2021 a las 10:59

6 registrados Guardar

'eduardo' tiene la palabra 'edu' y, aunque no forme parte de la extensión, lo da por válido porque contiene la cadena. Entonces aquí viene el segundo método, que se basa en separar el correo en partes mediante *splits* y obtener solamente la extensión:

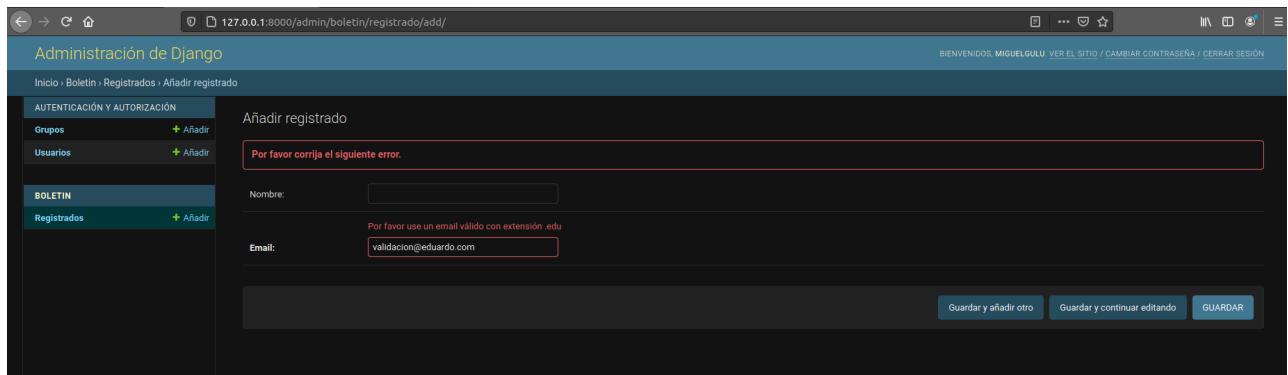


```

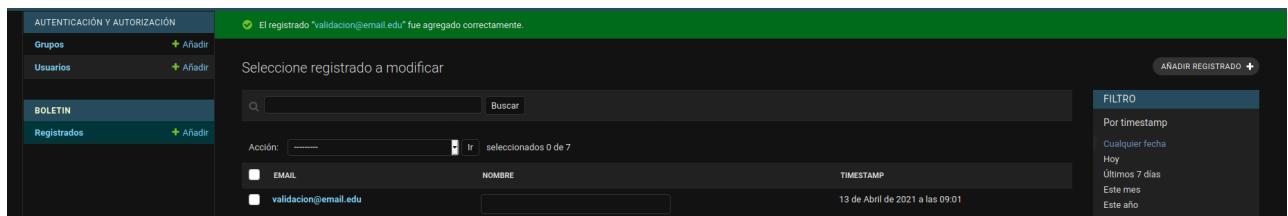
1  from django import forms
2  from .models import Registrado
3
4  class RegModelForm(forms.ModelForm):
5      class Meta:
6          model = Registrado
7          fields = ['nombre', 'email']
8
9      def clean_email(self):
10         email = self.cleaned_data.get("email")
11         email_base, proveedor = email.split("@")
12         dominio, extension = proveedor.split(".")
13         if not extension == "edu":
14             raise forms.ValidationError("Por favor use un email válido con extensión .edu")
15         return email
16
17 class RegForm(forms.Form):
18     nombre = forms.CharField(max_length=100)
19     email = forms.EmailField()
20

```

Con esto, ahora si tenemos una forma más eficiente de validar nuestro *email*:



Si escribimos la extensión '.edu' nos dará el registro como válido y se nos guardará.



Para terminar esta práctica, vamos a hacer lo mismo que hicimos para registrar el *email* pero con el nombre...

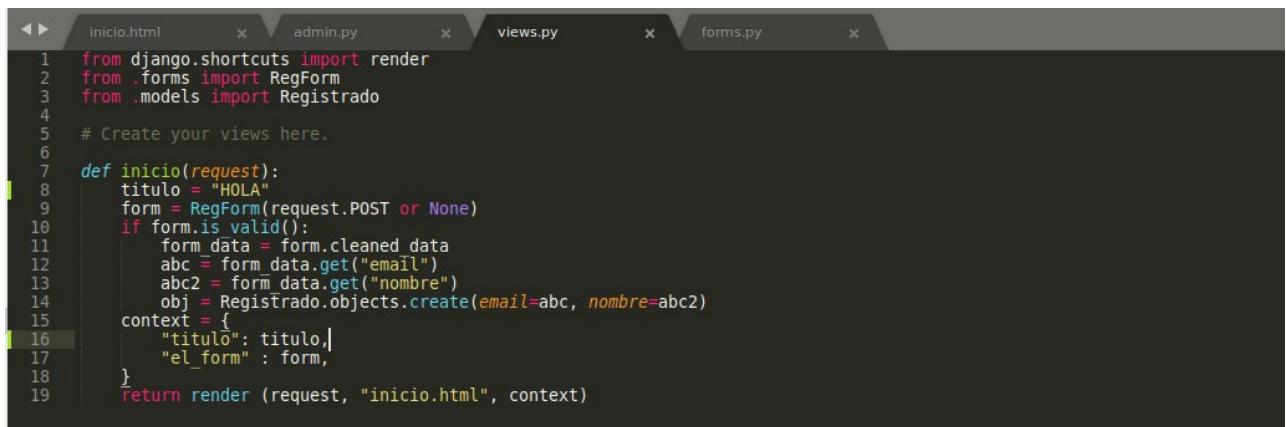


A screenshot of a code editor showing a Python file named 'forms.py'. The file contains code for Django forms, specifically for a 'Registrado' model. It includes a custom form class 'RegModelForm' that validates email addresses to ensure they end in '.edu'. It also includes a standard form class 'RegForm' with fields for name and email.

```
1 from django import forms
2
3 from .models import Registrado
4
5 class RegModelForm(forms.ModelForm):
6     class Meta:
7         model = Registrado
8         fields = ['nombre', 'email']
9
10    def clean_email(self):
11        email = self.cleaned_data.get("email")
12        email_base, proveedor = email.split("@")
13        dominio, extension = proveedor.split(".")
14        if not extension == "edu":
15            raise forms.ValidationError("Por favor use un email válido con extensión .edu")
16        return email
17
18    def clean_nombre(self):
19        nombre = self.cleaned_data.get("nombre")
20        #validacione|
21        return nombre
22
23 class RegForm(forms.Form):
24     nombre = forms.CharField(max_length=100)
25     email = forms.EmailField()
```

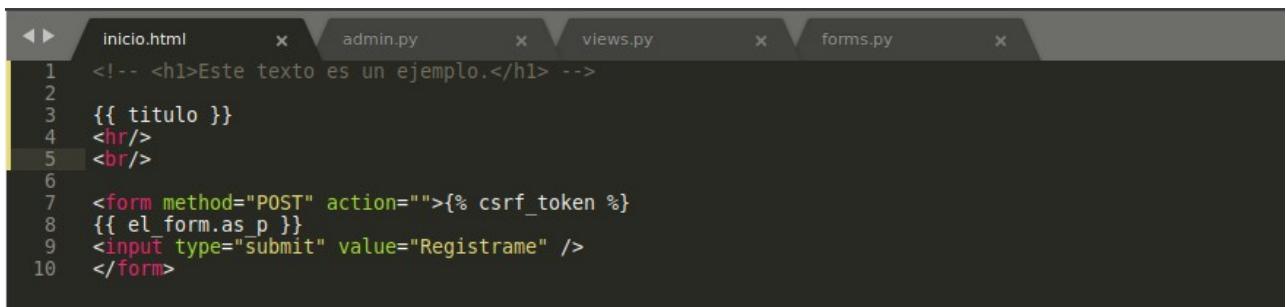
18. CONTEXTO VISTA

En esta parte del curso vamos a aprender a darle un poco más de contexto a nuestra vista en cuanto al formulario que tenemos. Para ello, vamos a nuestro ‘views.py’ y vamos a crear una variable nueva que la vamos a poner como título y la cual hay que añadir a *context*:



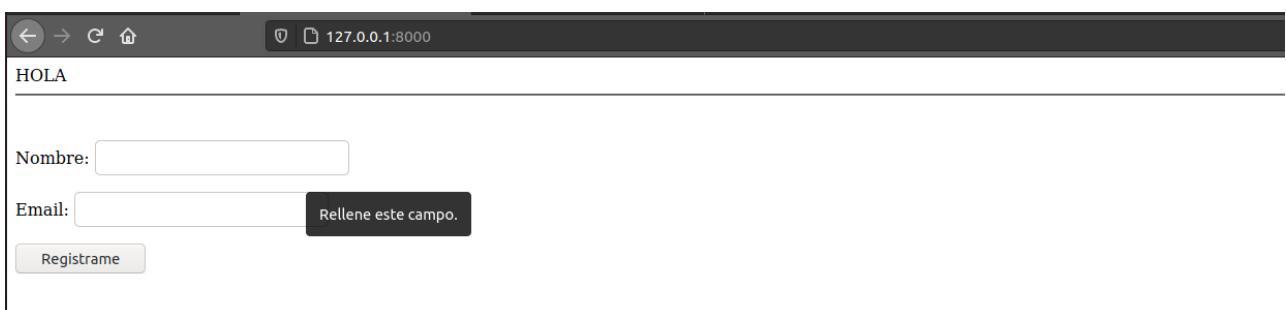
```
1 from django.shortcuts import render
2 from .forms import RegForm
3 from .models import Registrado
4
5 # Create your views here.
6
7 def inicio(request):
8     titulo = "HOLA"
9     form = RegForm(request.POST or None)
10    if form.is_valid():
11        form_data = form.cleaned_data
12        abc = form_data.get("email")
13        abc2 = form_data.get("nombre")
14        obj = Registrado.objects.create(email=abc, nombre=abc2)
15        context = {
16            "titul0": titulo,
17            "el_form": form,
18        }
19    return render (request, "inicio.html", context)
```

Para poder verlo en nuestra vista debemos añadir esa variable a la plantilla.

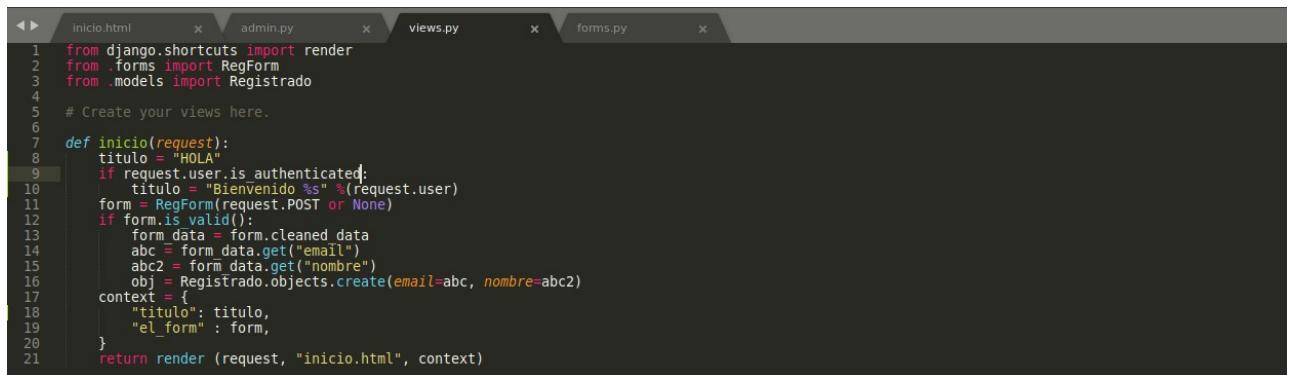


```
1 <!-- <h1>Este texto es un ejemplo.</h1> -->
2
3 {{ titulo }}
4 <hr/>
5 <br/>
6
7 <form method="POST" action="">{% csrf_token %}
8 {{ el_form.as_p }}
9 <input type="submit" value="Registrate" />
10 </form>
```

En nuestra página de inicio se vería algo tal que así:



A continuación vamos a hacer unos pequeños cambios para que la vista se ajuste al usuario que está logueado en la aplicación. Para ello, vamos a añadir una condición a nuestra vista de forma que, si el usuario esta registrado, la vista muestre un nuevo título. El código para ello sería lo siguiente..

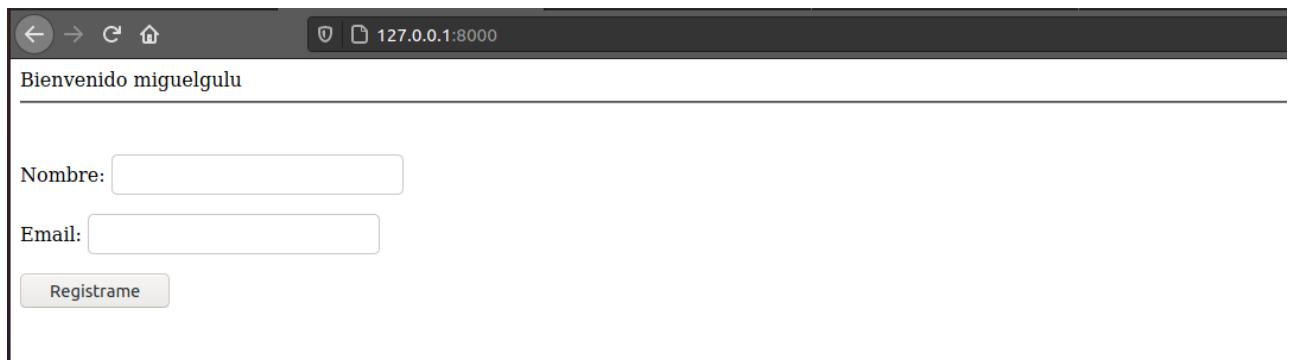


```

1  from django.shortcuts import render
2  from .forms import RegForm
3  from .models import Registrado
4
5  # Create your views here.
6
7  def inicio(request):
8      titulo = "HOLA"
9      if request.user.is_authenticated:
10          titulo = "Bienvenido %s" %(request.user)
11      form = RegForm(request.POST or None)
12      if form.is_valid():
13          form_data = form.cleaned_data
14          abc = form_data.get("email")
15          abc2 = form_data.get("nombre")
16          obj = Registrado.objects.create(email=abc, nombre=abc2)
17      context = {
18          "titulo": titulo,
19          "el_form": form,
20      }
21      return render (request, "inicio.html", context)

```

Entonces al recargar la página, saldría esto:



Pero si abrimos una pestaña de incógnito, donde no tenemos la sesión iniciada, se mantendrá el mensaje original.



Vamos a probar otras opciones para darle otro formato.

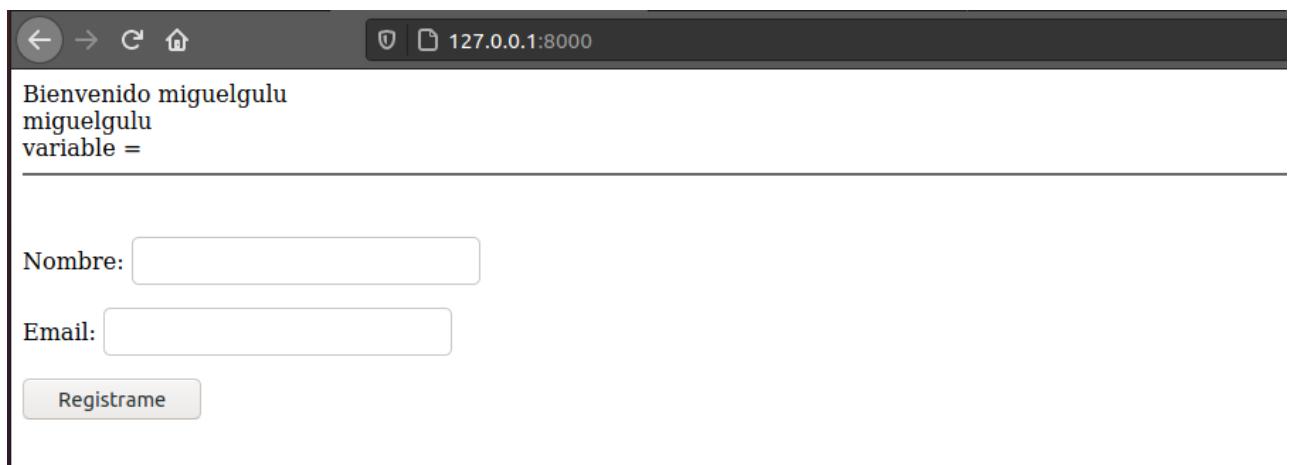


```

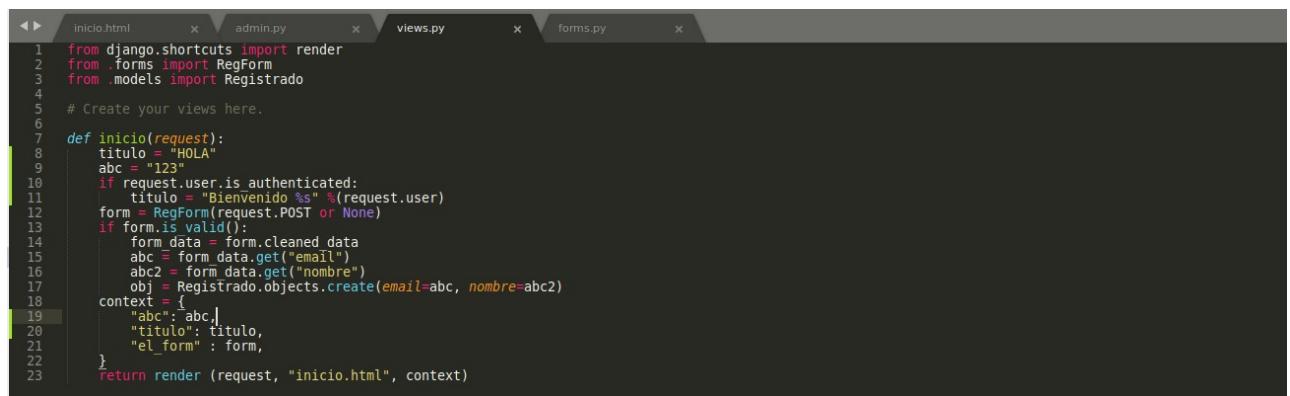
1 <!-- <h1>Este texto es un ejemplo.</h1> -->
2
3 {{ titulo }}<br/>
4 {{ request.user }}<br/>
5 variable = {{ abc }}
6 <hr/>
7 <br/>
8
9 <form method="POST" action="">{{ csrf_token }}
10 {{ el_form.as_p }}
11 <input type="submit" value="Registrate" />
12 </form>

```

Con esto, la vista mostrará el título, el usuario que está autenticado y una variable que en principio no aparece ya que no está declarada en ese entonces.



Si queremos que aparezca algo, tendremos que declararla:



```

1 from django.shortcuts import render
2 from .forms import RegForm
3 from .models import Registrado
4
5 # Create your views here.
6
7 def inicio(request):
8     titulo = "HOLA"
9     abc = "123"
10    if request.user.is_authenticated:
11        titulo = "Bienvenido %s" %(request.user)
12        form = RegForm(request.POST or None)
13        if form.is_valid():
14            form_data = form.cleaned_data
15            abc = form_data.get("email")
16            abc2 = form_data.get("nombre")
17            obj = Registrado.objects.create(email=abc, nombre=abc2)
18        context = {
19            "abc": abc,
20            "titulo": titulo,
21            "el_form": form,
22        }
23    return render (request, "inicio.html", context)

```

Entonces este es el resultado:

Bienvenido miguelgulu
miguelgulu
variable = 123

Nombre:

Email:

Bienvenido miguelgulu
miguelgulu

Nombre:

Email:

19. MODEL FORM EN LA VISTA

Para darle una mejor estructuración a nuestra vista, vamos a cambiar el ModelForm. Primero vamos a dar la opción de guardar objetos según el modelo, por lo que lo importamos; quitamos todas las variables que pusimos para guardar los datos y añadimos una nueva que contenga el guardado de todo según el modelo:

```

from django.shortcuts import render
from forms import RegForm, RegModelForm
from models import Registrado

# Create your views here.

def inicio(request):
    titulo = "HOLA"
    abc = "123"
    if request.user.is_authenticated:
        titulo = "Bienvenido %s" %(request.user)
    form = RegModelForm(request.POST or None)
    if form.is_valid():
        instance = form.save(commit=False)
        print (instance)
        #form.data = form.cleaned_data
        #abc = form.data.get("email")
        #abc2 = form.data.get("nombre")
        #obj = Registrado.objects.create(email=abc, nombre=abc2)
    context = {
        "abc": abc,
        "titulo": titulo,
        "el_form": form,
    }
    return render (request, "inicio.html", context)

```

Por lo pronto todo sigue igual, pero al cambiar lo que teníamos y añadimos un registro nuevo:

```

System check identified no issues (0 silenced).
April 13, 2021 - 18:02:07
Django version 3.2, using settings 'pd110.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
[13/Apr/2021 18:02:11] "GET / HTTP/1.1" 200 531
Not Found: /favicon.ico
[13/Apr/2021 18:03:13] "POST / HTTP/1.1" 200 653
hola@test.edu
[13/Apr/2021 18:03:38] "POST / HTTP/1.1" 200 566

```

Como podemos ver se nos guarda el correo que introdujimos. El nombre no aparece ya que en el modelo en el return solo tenía el email.

Ahora podemos añadirle un timestamp al guardado:

```
instance = form.save(commit=False)
print (instance)
print (instance.timestamp)
#form_data = form.cleaned_data
```

Pero si registramos un nuevo usuario nos sale *None* ya que en el formulario no hemos completado ese campo y por lo cual, no existe. Todo es por que hay **commit False** (más abajo se puede ver)

```
[13/Apr/2021 18:05:31] "POST / HTTP/1.1" 200 566
hola@test.edu
None
[13/Apr/2021 18:05:38] "POST / HTTP/1.1" 200 566
```

```
if form.is_valid():
    instance = form.save(commit=False)
    instance.save()
    print (instance)
    print (instance.timestamp)
```

Con esto si se guardará los cambios y se registrará la fecha de registro:

```
[13/Apr/2021 18:06:43] "POST / HTTP/1.1" 200 566
hola@test.edu
2021-04-13 18:06:44.920642+00:00
[13/Apr/2021 18:06:44] "POST / HTTP/1.1" 200 566
```

Y todo está registrado en nuestro *admin*:

EMAIL	NOMBRE	TIMESTAMP
hola@test.edu	hola	13 de Abril de 2021 a las 18:06
hola@test.edu	hola	13 de Abril de 2021 a las 09:01
validacion@email.edu		13 de Abril de 2021 a las 08:57
validacion@eduardo.com		13 de Abril de 2021 a las 08:52
email@email.com		11 de Abril de 2021 a las 10:51
test2@gmail.com	test2	11 de Abril de 2021 a las 10:43
test@gmail.com		9 de Abril de 2021 a las 10:50
test@gmail.com	test	9 de Abril de 2021 a las 10:39
palotes@gmail.com	pepito	9 de Abril de 2021 a las 10:39

La idea de guardar los cambios mediante `instance.save()` y no cambiando el commit de False a True es que, según lo que registremos, pueda pasar una cosa u otra, vamos a poner el siguiente ejemplo:

```

11     titulo = "Bienvenido %s" %(request.user)
12     form = RegModelForm(request.POST or None)
13     if form.is_valid():
14         instance = form.save(commit=False)
15         if not instance.nombre:
16             instance.nombre = "Usuario"
17         instance.save()
18         print (instance)
19         print (instance.timestamp)

```

Vamos a poner que, como el campo de usuario no es válido, si el nombre está vacío, vamos a poder como nombre uno predeterminado.

EMAIL	NOMBRE	TIMESTAMP
usuario@usuario.edu	Usuario	13 de Abril de 2021 a las 18:09

Y podemos añadir una línea la cual, una vez que nos registremos, nos salte un nuevo mensaje con el usuario que se ha registrado:

```

1  from django.shortcuts import render
2  from .forms import RegForm, RegModelForm
3  from .models import Registrado
4
5  # Create your views here.
6  def inicio(request):
7      titulo = "MOLA"
8      abc = "123"
9      if request.user.is_authenticated:
10          titulo = "Bienvenido %s" %(request.user)
11      form = RegModelForm(request.POST or None)
12
13      context = {
14          "abc": abc,
15          "titulo": titulo,
16          "el_form": form,
17      }
18
19      if form.is_valid():
20          instance = form.save(commit=False)
21          if not instance.nombre:
22              instance.nombre = "Usuario"
23          instance.save()
24
25          context = {
26              "titulo": "Gracias por registrarte %s!" %(instance.nombre)
27          }
28
29
30          print (instance)
31          print (instance.timestamp)
32          #form_data = form.cleaned_data
33          #abc = form_data.get("email")
34          #abc2 = form_data.get("nombre")
35          #abc = Registrado.objects.create(email=abc, nombre=abc2)
36
37      return render (request, "inicio.html", context)

```

Introducimos un nuevo registro...

The screenshot shows a web browser window with the URL 127.0.0.1:8000. The page displays a success message: "Gracias por registrarte juanito! miguelgulu". Below this is a "Regístrate" button. At the bottom, there is a table with three columns: EMAIL, NOMBRE, and TIMESTAMP. The data in the table is:

EMAIL	NOMBRE	TIMESTAMP
<input checked="" type="checkbox"/> juanito@alpargata.edu	juanito	13 de Abril de 2021 a las 18:17

Y ahí tenemos el mensaje que saldría. Si no introducimos el nombre nos sale el siguiente:

The screenshot shows a web browser window with the URL 127.0.0.1:8000. The page displays a success message: "Gracias por registrarte Usuario! miguelgulu". Below this is a "Regístrate" button. At the bottom, there is a table with three columns: EMAIL, NOMBRE, and TIMESTAMP. The data in the table is:

EMAIL	NOMBRE	TIMESTAMP
<input checked="" type="checkbox"/> juanito@alpargata.edu	juanito	13 de Abril de 2021 a las 18:17

20. CUSTOM FORM PARA CONTACTO

A continuación, veremos como ajustar nuestro formulario y cambiarlo a un formulario de contacto. Entonces nos vamos a ‘forms.py’ y vamos a cambiar RegForm por uno de contacto:

```
class ContactForm(forms.Form):
    nombre = forms.CharField()
    email = forms.EmailField()
    mensaje = forms.CharField(widget=forms.Textarea)
```

Añadiremos un nuevo campo el cual contendrá el mensaje. Ese campo tendrá un widget el cual hará que se extienda el área donde escribiremos. Además de esto, de ‘views.py’ tendremos que borrar la importación ya que RegForm no existe y cambiarlo por la nueva clase.

También habrá que crear una función en las vistas para el contacto, el cuál será así:

```
39 def contact(request):
40     form = ContactForm(request.POST or None)
41     context = {
42         "form": form,
43     }
44     return render(request, "forms.html", context)
```

Para este nuevo formulario tendremos que crear una nueva plantilla, por lo cual, en la carpeta ‘templates’ añadimos una plantilla nueva:

```
<{{ titulo }}><br/>
<{{ request.user }}><br/>
<br/>
<br/>
<form method="POST" action="">{% csrf_token %}
<hr>
{{ form.as_p }}
<input type="submit" value="Regístrate" />
</form>
```

Y habrá que añadir la URL...

```
22 urlpatterns = [
23     path('admin/', admin.site.urls),
24     path('', views.inicio, name='inicio'),
25     path('contact/', views.contact, name='contact')
26 ]
27
```

Al entrar finalmente en nuestra URL tenemos nuestro formulario de contacto.

miguelgulu

Nombre:

Email:

Mensaje:

Regístrate

Todos estos campos son obligatorios, ya que en el modelo no tenemos puesto nada sobre ellos. Habrá que añadirlo al igual que el anterior. Vamos a poner que es obligatorio el campo de *email*, ya que se necesita de él para comunicarse con quién nos contacto:

```
23 class ContactForm(forms.Form):  
24     nombre = forms.CharField(required=False)  
25     email = forms.EmailField()  
26     mensaje = forms.CharField(widget=forms.Textarea)
```

miguelgulu

Nombre:

Email:

Mensaje:

Regístrate

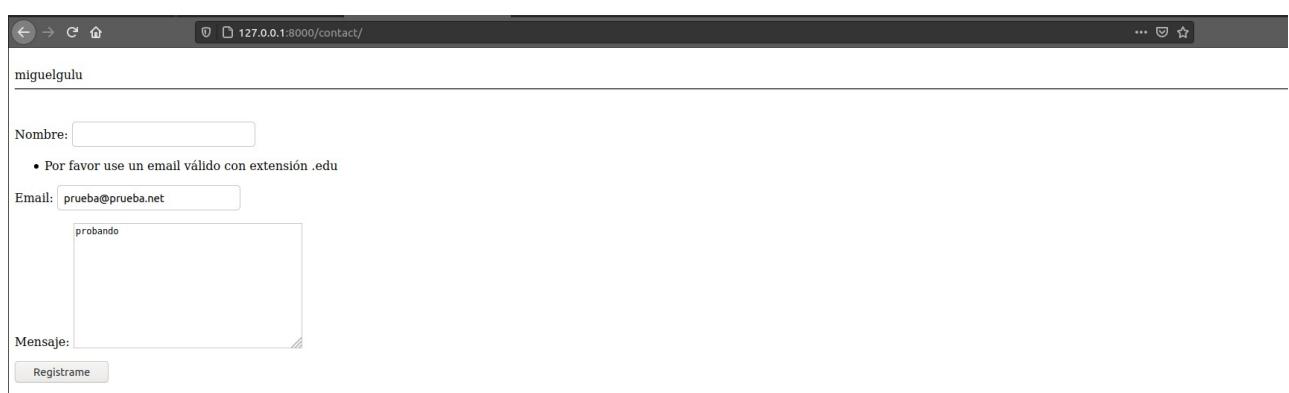
Vamos a añadir una condición de que si el formulario es válido, devuelva por pantalla los campos limpios:

```
39     def contact(request):
40         form = ContactForm(request.POST or None)
41         if form.is_valid():
42             print (form.cleaned_data)
43             context = {
44                 "form": form,
45             }
46             return render(request, "forms.html", context)
```

Y vamos a añadir el método de validación de extensión al igual que en el formulario anterior:

```
23     class ContactForm(forms.Form):
24         nombre = forms.CharField(required=False)
25         email = forms.EmailField()
26         mensaje = forms.CharField(widget=forms.Textarea)
27
28     def clean_email(self):
29         email = self.cleaned_data.get("email")
30         email_base, proveedor = email.split("@")
31         dominio, extension = proveedor.split(".")
32         if not extension == "edu":
33             raise forms.ValidationError("Por favor use un email válido con extensión .edu")
34         return email
```

Vamos a hacer la prueba:



Nos salta el aviso de que el email introducido no es válido. Una vez probado, vamos a hacer unos cambios más y añadir cada valor introducido en cada campo a una variable:

```

39  def contact(request):
40      form = ContactForm(request.POST or None)
41      if form.is_valid():
42          nombre = form.cleaned_data.get("nombre")
43          email = form.cleaned_data.get("email")
44          mensaje = form.cleaned_data.get("mensaje")
45          print(email, mensaje, nombre)
46          context = {
47              "form": form,
48          }
49      return render(request, "forms.html", context)

```

Ahora toca hacer la prueba en el formulario...

miguelgulu

Nombre:

- Por favor use un email válido con extensión .edu

Email:

Mensaje:

```

[15/Apr/2021 10:09:27] "POST /contact/ HTTP/1.1" 200 685
prueba@prueba.edu probando prueba
[15/Apr/2021 10:10:11] "POST /contact/ HTTP/1.1" 200 613

```

En el terminal se nos devuelve los campos que hemos rellenado.

Finalmente vamos a hacer igual que en el modelo anterior y vamos a quitar las variables.

```

39  def contact(request):
40      form = ContactForm(request.POST or None)
41      if form.is_valid():
42          for key in form.cleaned_data:
43              print(key)
44              print(form.cleaned_data.get(key))
45          #nombre = form.cleaned_data.get("nombre")
46          #email = form.cleaned_data.get("email")
47          #mensaje = form.cleaned_data.get("mensaje")
48          #print(email, mensaje, nombre)
49          context = {
50              "form": form,
51          }
52      return render(request, "forms.html", context)

```

Introducimos los mismos datos en el formulario y, como podremos ver, los campos que hemos rellenado van separados.

```
[15/Apr/2021 10:16:26] "POST /contact/ HTTP/1.1" 200 613
nombre
prueba
email
prueba@prueba.edu
mensaje
probando
[15/Apr/2021 10:16:34] "POST /contact/ HTTP/1.1" 200 613
```

Otra forma de hacerlo puede ser así:

```
39     def contact(request):
40         form = ContactForm(request.POST or None)
41         if form.is_valid():
42             for key, value in form.cleaned_data.items():
43                 print(key, value)
44             #for key in form.cleaned_data:
45             #    print(key)
46             #    print(form.cleaned_data.get(key))
47             #nombre = form.cleaned_data.get("nombre")
48             #email = form.cleaned_data.get("email")
49             #mensaje = form.cleaned_data.get("mensaje")
50             #print(email, mensaje, nombre)
51             context = {
52                 "form": form,
53             }
54         return render(request, "forms.html", context)
```

Karlita al usar Python 2 usa la función ‘iteritems’ pero, en Python 3, la función ‘items’ ya contiene ‘iteritems’. El resto sigue igual.

```
[15/Apr/2021 10:23:19] "GET /contact/ HTTP/1.1" 200 564
nombre prueba
email prueba@prueba.edu
mensaje nueva prueba
[15/Apr/2021 10:23:50] "POST /contact/ HTTP/1.1" 200 617
```

21. CONFIGURAR EMAIL

En este vídeo vamos a aprender a configurar nuestro email de la aplicación. Para ello, hay que establecer una serie de parámetros en los *settings* que son los siguientes:

```

26 DEBUG = True
27
28 ALLOWED_HOSTS = []
29
30 EMAIL_HOST = 'smtp.gmail.com'
31 EMAIL_HOST_USER = 'tu_email@gmail.com'
32 EMAIL_HOST_PASSWORD = 'tupassword'
33 EMAIL_PORT = 587
34 EMAIL_USE_TLS = True
35
36 # Application definition
37
38 INSTALLED_APPS = [
39     'django.contrib.admin',

```

Ahora tendremos que ir a nuestra vista y quitamos el comentario en las variables definidas en la función: *settings*

```

def contact(request):
    form = ContactForm(request.POST or None)
    if form.is_valid():
        for key, value in form.cleaned_data.items():
            print(key, value)
        #for key in form.cleaned_data:
        #    print(key)
        #    print(form.cleaned_data.get(key))
        nombre = form.cleaned_data.get("nombre")
        email = form.cleaned_data.get("email")
        mensaje = form.cleaned_data.get("mensaje")
        #print(email, mensaje, nombre)
    context = {
        "form": form,
    }
    return render(request, "forms.html", context)

```

E importamos este módulo de django:

```

1 from django.shortcuts import render
2 from django.core.mail import send_mail

```

Y los *settings*:

```

2 from django.core.mail import send_mail
3 from django.conf import settings
4

```

Y dentro de nuestra función de ‘contact’ tendremos que meter una serie de parámetros que definen el envío de emails:

```

42     def contact(request):
43         form = ContactForm(request.POST or None)
44         if form.is_valid():
45             for key, value in form.cleaned_data.items():
46                 print (key, value)
47             #for key in form.cleaned_data:
48             #    print (key)
49             #    print(form.cleaned_data.get(key))
50             nombre = form.cleaned_data.get("nombre")
51             email = form.cleaned_data.get("email")
52             mensaje = form.cleaned_data.get("mensaje")
53             #print(email, mensaje, nombre)
54             send_mail(asunto,
55                       mensaje_email,
56                       email_from,
57                       [email_to],
58                       fail_silently=False
59                     )
60             context = {
61                 "form": form,
62             }
63             return render(request, "forms.html", context)

```

Además, tendremos que añadir como variables los campos que lo van a formar...

```

def contact(request):
    form = ContactForm(request.POST or None)
    if form.is_valid():
        for key, value in form.cleaned_data.items():
            print (key, value)
        #for key in form.cleaned_data:
        #    print (key)
        #    print(form.cleaned_data.get(key))
        form_nombre = form.cleaned_data.get("nombre")
        form_email = form.cleaned_data.get("email")
        form_mensaje = form.cleaned_data.get("mensaje")
        asunto = 'Formulario de Contacto'
        email_from = settings.EMAIL_HOST_USER
        email_to = [email_from, "test@gmail.com"]
        mensaje_email = "%s: %s enviado por %s" %(form_nombre, form_mensaje, form_email)
        #print(email, mensaje, nombre)
        send_mail(asunto,
                  mensaje_email,
                  email_from,
                  [email_to],
                  fail_silently=False
                )
    context = {
        "form": form,
    }
    return render(request, "forms.html", context)

```

Recargamos nuestra página de contacto y rellenamos el formulario:

miguelgulu

Nombre: alunso

Email: probando@gmail.com

Mensaje:
hola!

Regístrate

SMTPAuthenticationError at /contact/

(535, b'5.7.8 Username and Password not accepted. Learn more at\n5.7.8 https://support.google.com/mail/?p=BadCredentials u8sm9545949wrr.42 - gsmtp')

Request Method: POST
Request URL: http://127.0.0.1:8000/contact/
Django Version: 3.2
Exception Type: SMTPAuthenticationError
Exception Value: (535, b'5.7.8 Username and Password not accepted. Learn more at\n5.7.8 https://support.google.com/mail/?p=BadCredentials u8sm9545949wrr.42 - gsmtp')
Exception Location: /usr/lib/python3.8/smtplib.py, line 646, in auth
Python Executable: /home/alunso/Documentos/LM/cursos/karlita/venv/bin/python3
Python Version: 3.8.5
Python Path: ['/home/alunso/Documentos/LM/cursos/karlita/src', '/usr/lib/python38.zip', '/usr/lib/python3.8', '/usr/lib/python3.8/lib-dynload', '/home/alunso/Documentos/LM/cursos/karlita/venv/lib/python3.8/site-packages']
Server time: Fri, 16 Apr 2021 10:15:21 +0000

Traceback [Switch to copy-and-paste view](#)

```
/home/alunso/Documentos/LM/cursos/karlita/venv/lib/python3.8/site-packages/django/core/handlers/exception.py, line 47, in inner
    response = get_response(request)
▶ Local vars

/home/alunso/Documentos/LM/cursos/karlita/venv/lib/python3.8/site-packages/django/core/handlers/base.py, line 181, in _get_response
    response = wrapped_callback(request, *callback_args, **callback_kwargs)
▶ Local vars

/home/alunso/Documentos/LM/cursos/karlita/src/boletin/views.py, line 58, in contact
    send_mail(asunto,
```

Y nos sale ese error porque el correo electrónico que hemos especificado no es real y ha intentado contactar con google para comunicarse con ese correo. Si cambiaramos ese ‘*fail_silently = False*’ a **True**, no nos saldría ese error pero no nos comunicaría de que no es posible.

22. CONFIGURACIÓN DE ARCHIVOS ESTÁTICOS

A partir de este vídeo nos dedicaremos más a configuraciones de diseño. Empezaremos con los archivos estáticos en un entorno de desarrollo. Estos archivos son como por ejemplo CSS y JavaScript.

Primero debemos empezar asegurándonos que en *INSTALLED_APPS* tenemos *staticfiles* añadido:

```
8 INSTALLED_APPS = [
9     'django.contrib.admin',
10    'django.contrib.auth',
11    'django.contrib.contenttypes',
12    'django.contrib.sessions',
13    'django.contrib.messages',
14    'django.contrib.staticfiles',
15    'boletin.apps.BoletinConfig'
16]
```

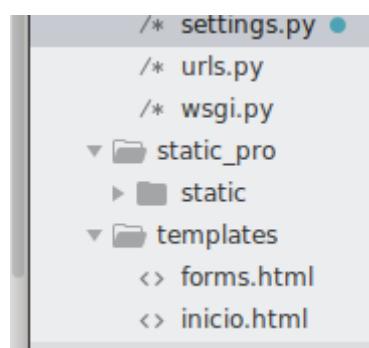
Y STATIC_URL:

```
123 # Static files (CSS, JavaScript, Images)
124 # https://docs.djangoproject.com/en/3.2/howto/static-files/
125
126 STATIC_URL = '/static/'
127
```

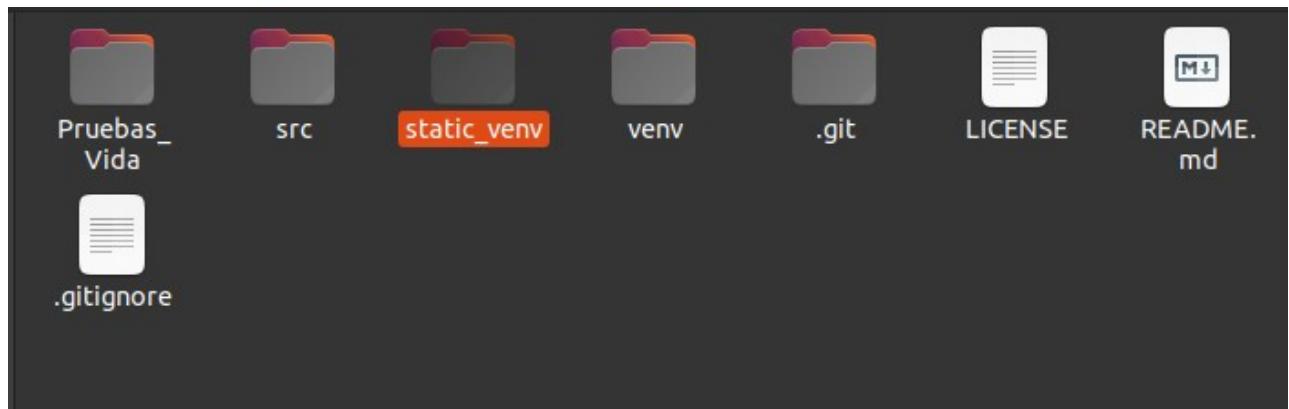
Y debemos añadir el directorio donde se localizarán nuestros archivos estáticos. En Python 3 es distinto a Python 2, y tiene la siguiente estructura:

```
127
128 #/static/imagenes/img1.jpg
129
130 STATICFILES_DIRS = [
131     BASE_DIR / "static_pro" / "static",
132     #'/var/www/static/',
133 ]
134
135 # Default primary key field type
```

Por lo cual tenemos que tener creadas las carpetas correspondientes:



Y debemos crear una carpeta para hacer la imitación de que tenemos los archivos en producción en otro servidor.



Además hace falta especificar STATIC_ROOT donde vivirán nuestros archivos en otro servidor:

```
133     ]
134
135 STATIC_ROOT = "/home/alunsiito/Documentos/LM/cursos/karlita/static_env/static_root"
136
137 # Default primary key field type
```

También podemos añadir directorios de archivos estáticos media:

```
125
126 STATIC_URL = '/static/'
127 STATIC_URL = '/media/'
128
129 #static/imagenes/img1.jpg
130
131 STATICFILES_DIRS = [
132     BASE_DIR / "static_pro" / "static",
133     "/var/www/static/",
134 ]
135
136 STATIC_ROOT = "/home/alunsiito/Documentos/LM/cursos/karlita/static_env/static_root"
137
138 MEDIA_ROOT = "/home/alunsiito/Documentos/LM/cursos/karlita/static_env/media_root"
139
```

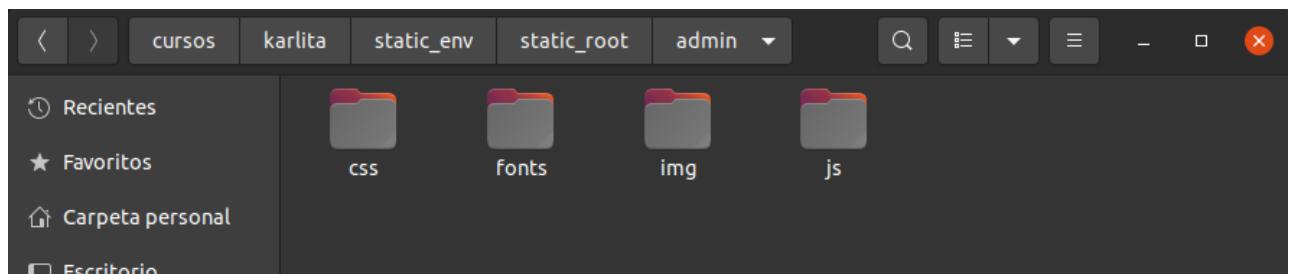
Finalmente nos quedaría añadir la url:

```
16 from django.contrib import admin
17 from django.urls import path
18
19 from django.conf import settings
20 from django.conf.urls.static import static
21
22 from boletin import views
23
24
25 urlpatterns = [
26     path('admin/', admin.site.urls),
27     path('', views.inicio, name='inicio'),
28     path('contact/', views.contact, name='contact')
29 ]
30
31 if setting.DEBUG:
32     urlpatterns += static(settings.STATIC_URL, document_root=settings.STATIC_ROOT)
33     urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
34
```

Y si ejecutamos el siguiente comando para enviar nuestros archivos estáticos al servidor:

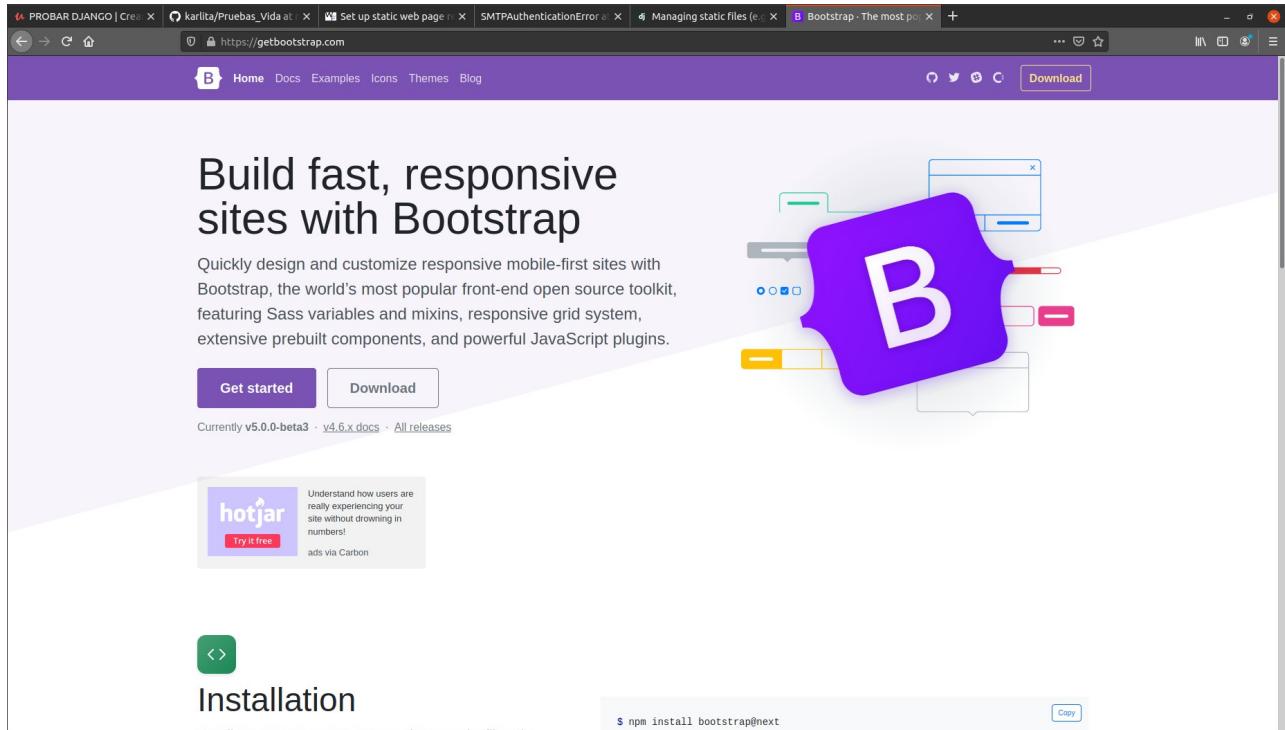
```
(venv) alunsiito@alunsiito:~/Documentos/LM/cursos/karlita/src$ python3 manage.py collectstatic
128 static files copied to '/home/alunsiito/Documentos/LM/cursos/karlita/static_env/static_root'.
(venv) alunsiito@alunsiito:~/Documentos/LM/cursos/karlita/src$
```

Y como vemos, en la carpeta de static_root nos aparece una nueva carpeta, la cual contiene todos los archivos que hemos enviado:

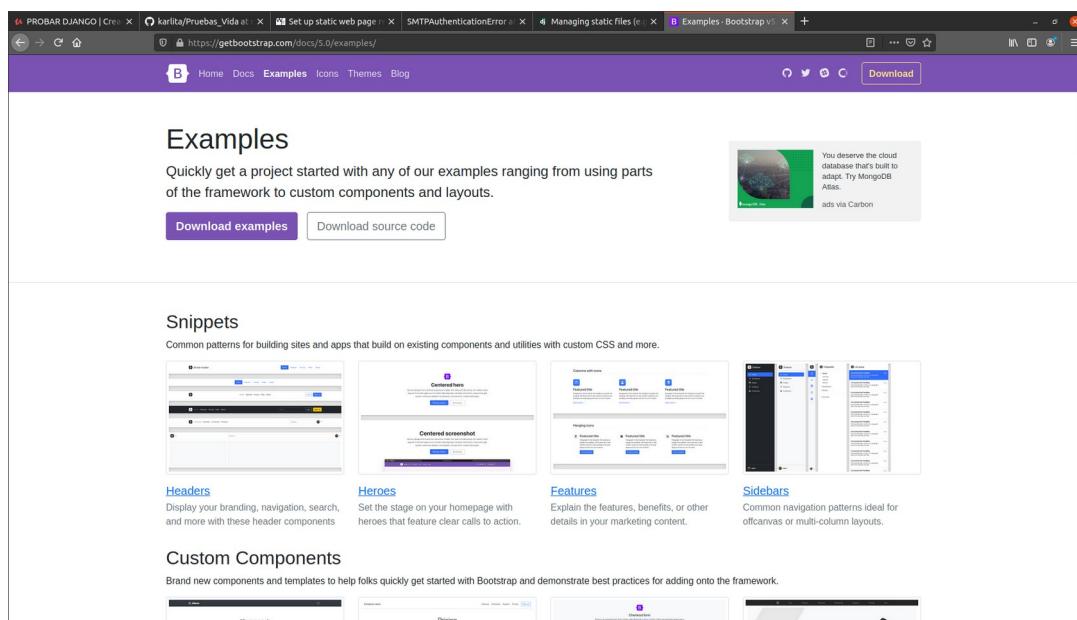


23. CONFIGURACIÓN BOOTSTRAP

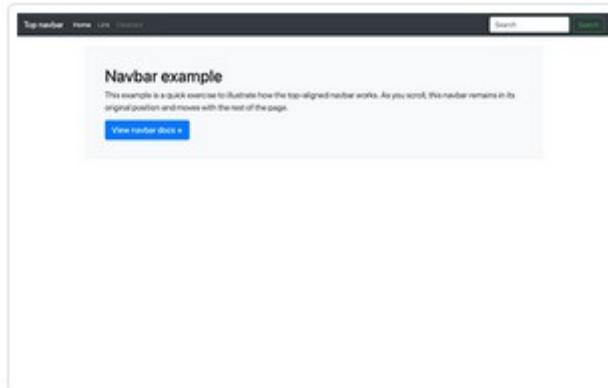
En esta práctica vamos a trabajar con Bootstrap, el cual es un framework de HTML, CSS y JavaScript y nos permite cosas como añadir elementos responsivos al nuestra página.



Nosotros vamos a usar una plantilla que tienen ya prediseñado antes que usar el Boostrap CDN, el cual lo que hace es reconocer elementos ya guardados.



Si buscamos en ejemplos vamos a tener varias plantillas que podemos usar. Nosotros vamos a usar una que se llama “*Navbar static*”:



Navbar static

Single navbar example of a static top navbar along with some additional content.

Hacemos click y, al pulsar click derecho para ver el código fuente podremos ver de que se compone y como funciona:

```

<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta name="description" content="">
    <meta name="author" content="Mark Otto, Jacob Thornton, and Bootstrap contributors">
    <meta name="generator" content="Hugo 0.82.0">
    <title>Top navbar example · Bootstrap v5.0.4</title>
    <link rel="canonical" href="https://getbootstrap.com/docs/5.0/examples/navbar-static/">

    <!-- Bootstrap core CSS -->
    <link href="/docs/5.0/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-eOJMyrsd53li+sC0/bJGFsiCz+5NDVN2yr8+0RDqj00l0h+rP48ckxlpbkGwra6" crossorigin="anonymous">
    <!-- Favicons -->
    <link rel="apple-touch-icon" href="/docs/5.0/assets/img/favicons/apple-touch-icon.png" sizes="180x180">
    <link rel="icon" href="/docs/5.0/assets/img/favicons/favicon-32x32.png" sizes="32x32" type="image/png">
    <link rel="icon" href="/docs/5.0/assets/img/favicons/favicon-16x16.png" sizes="16x16" type="image/png">
    <link rel="manifest" href="/docs/5.0/assets/img/favicons/manifest.json">
    <link rel="mask-icon" href="/docs/5.0/assets/img/favicons/safari-pinned-tab.svg" color="#7952b3">
    <link rel="icon" href="/docs/5.0/assets/img/favicons/favicon.ico">
    <meta name="theme-color" content="#7952b3">

    <style>
      .bd-placeholder-img {
        font-size: 1.25rem;
        text-anchor: middle;
        -webkit-user-select: none;
        -moz-user-select: none;
        user-select: none;
      }

      @media (min-width: 768px) {
        .bd-placeholder-img-lg {
          font-size: 3.5rem;
        }
      }
    </style>

    <!-- Custom styles for this template -->
    <link href="navbar-top.css" rel="stylesheet">
  </head>
  <body>

    <nav class="navbar navbar-expand-md navbar-dark bg-dark mb-4">
      <div class="container-fluid">
        <a class="navbar-brand" href="#">Top navbar</a>
        <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarCollapse" aria-controls="navbarCollapse" aria-expanded="false" aria-label="Toggle navigation">
          <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarCollapse">
          <div class="navbar-nav me-auto mb-2 mb-md-0">
            <li class="nav-item">
              <a class="nav-link active" aria-current="page" href="#">Home</a>
            </li>
            <li class="nav-item">
              <a class="nav-link" href="#">Link</a>
            </li>
            <li class="nav-item">
              <a class="nav-link disabled" href="#" tabindex="-1" aria-disabled="true">Disabled</a>
            </li>
          </div>
        </div>
      </div>
    </nav>
  </body>
</html>

```

Vamos a guardar esa plantilla (en la carpeta templates) y la vamos a modificar...

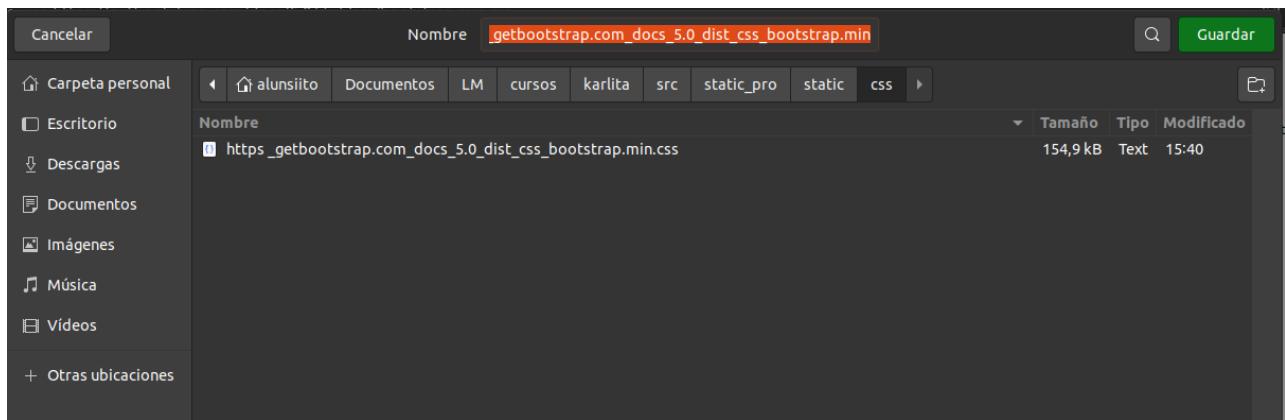
```

1 <!doctype html>
2 <html lang="en">
3   <head>
4     <meta charset="utf-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1">
6     <meta name="description" content="Top navbar example · Bootstrap v5.0 example" data-bs-globals="true">
7     <meta name="author" content="Mark Otto, Jacob Thornton, and Bootstrap contributors" data-bs-globals="true">
8     <meta name="generator" content="Hugo 0.82.0" data-bs-globals="true">
9     <title>Top navbar example · Bootstrap v5.0</title>
10    <link rel="canonical" href="https://getbootstrap.com/docs/5.0/examples/navbar-static/" data-bs-globals="true">
11
12    <!-- Bootstrap core CSS -->
13    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-EVSTQNDeOOGzJGvZBjwqfDyQWzXnLcCz+oPdRgq0l0h+rP48cklpbKgwra6" crossorigin="anonymous">
14
15    <!-- Favicons -->
16    <link rel="apple-touch-icon" href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/img/favicons/apple-touch-icon.png" sizes="180x180" data-bs-globals="true">
17    <link rel="icon" href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/img/favicons/favicon-32x32.png" sizes="32x32" type="image/png" data-bs-globals="true">
18    <link rel="icon" href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/img/favicons/favicon-16x16.png" sizes="16x16" type="image/png" data-bs-globals="true">
19    <link rel="mask-icon" href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/img/favicons/safari-pinned-tab.svg" color="#7952b3" data-bs-globals="true">
20    <link rel="icon" href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/img/favicons/favicon.ico" data-bs-globals="true">
21
22    <meta name="theme-color" content="#7952b3" data-bs-globals="true">
23
24
25
26
27
28    <style>
29      <.bd-placeholder-img {
30        font-size: 1.125rem;
31        text-anchor: middle;
32        -webkit-user-select: none;
33        -moz-user-select: none;
34        user-select: none;
35      }
36
37      @media (min-width: 768px) {
38        .bd-placeholder-img-lg {
39          font-size: 3.5rem;
40        }
41      }
42    </style>
43
44
45    <!-- Custom styles for this template -->
46    <link href="navbar-top.css" rel="stylesheet">
47  </head>
48  <body>
49
50    <nav class="navbar navbar-expand-md navbar-dark bg-dark mb-4">
51      <div class="container-fluid">
52        <a class="navbar-brand" href="#">Top navbar</a>
53        <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarCollapse" aria-controls="navbarCollapse" aria-expanded="false" aria-label="Toggle navigation">
54          <span class="navbar-toggler-icon"></span>
55        </button>
56        <div class="collapse navbar-collapse" id="navbarCollapse">
57          <ul class="navbar-nav me-auto mb-2 mb-md-0">
58            <li class="nav-item">
59              <a class="nav-link active" aria-current="page" href="#">Home</a>
60            </li>
61            <li class="nav-item">
62              <a class="nav-link" href="#">Link</a>
63            </li>
64          </ul>
65        </div>
66      </div>
67    </nav>
68
69
70
71
72
73
74
75
76
77
78
79
79
80
81
82
83
84
85
86
87
88
89
89
90
91
92
93
94
95
96
97
98
99
99
100
101
102
103
104
105
106
107
108
109
109
110
111
112
113
114
115
116
117
118
119
119
120
121
122
123
124
125
126
127
127
128
129
129
130
131
132
133
134
135
136
137
137
138
139
139
140
141
142
143
144
145
146
147
147
148
149
149
150
151
152
153
154
155
156
157
158
159
159
160
161
162
163
164
165
166
167
167
168
169
169
170
171
172
173
174
175
176
177
177
178
179
179
180
181
182
183
184
185
186
186
187
188
188
189
189
190
191
192
193
194
195
196
197
197
198
199
199
200
201
202
203
204
205
205
206
207
207
208
208
209
209
210
211
212
213
214
215
215
216
216
217
217
218
218
219
219
220
221
222
223
224
225
225
226
226
227
227
228
228
229
229
230
231
232
233
233
234
234
235
235
236
236
237
237
238
238
239
239
240
241
242
243
243
244
244
245
245
246
246
247
247
248
248
249
249
250
251
252
253
253
254
254
255
255
256
256
257
257
258
258
259
259
260
261
262
263
263
264
264
265
265
266
266
267
267
268
268
269
269
270
271
272
273
273
274
274
275
275
276
276
277
277
278
278
279
279
280
281
282
283
283
284
284
285
285
286
286
287
287
288
288
289
289
290
291
292
293
293
294
294
295
295
296
296
297
297
298
298
299
299
300
301
302
303
303
304
304
305
305
306
306
307
307
308
308
309
309
310
311
312
313
313
314
314
315
315
316
316
317
317
318
318
319
319
320
321
322
322
323
323
324
324
325
325
326
326
327
327
328
328
329
329
330
331
332
332
333
333
334
334
335
335
336
336
337
337
338
338
339
339
340
341
342
342
343
343
344
344
345
345
346
346
347
347
348
348
349
349
350
351
352
352
353
353
354
354
355
355
356
356
357
357
358
358
359
359
360
361
362
362
363
363
364
364
365
365
366
366
367
367
368
368
369
369
370
371
372
372
373
373
374
374
375
375
376
376
377
377
378
378
379
379
380
381
382
382
383
383
384
384
385
385
386
386
387
387
388
388
389
389
390
391
392
392
393
393
394
394
395
395
396
396
397
397
398
398
399
399
400
401
402
402
403
403
404
404
405
405
406
406
407
407
408
408
409
409
410
411
412
412
413
413
414
414
415
415
416
416
417
417
418
418
419
419
420
420
421
421
422
422
423
423
424
424
425
425
426
426
427
427
428
428
429
429
430
431
432
432
433
433
434
434
435
435
436
436
437
437
438
438
439
439
440
441
442
442
443
443
444
444
445
445
446
446
447
447
448
448
449
449
450
451
452
452
453
453
454
454
455
455
456
456
457
457
458
458
459
459
460
461
462
462
463
463
464
464
465
465
466
466
467
467
468
468
469
469
470
471
472
472
473
473
474
474
475
475
476
476
477
477
478
478
479
479
480
481
482
482
483
483
484
484
485
485
486
486
487
487
488
488
489
489
490
491
492
492
493
493
494
494
495
495
496
496
497
497
498
498
499
499
500
501
502
502
503
503
504
504
505
505
506
506
507
507
508
508
509
509
510
511
512
512
513
513
514
514
515
515
516
516
517
517
518
518
519
519
520
521
522
522
523
523
524
524
525
525
526
526
527
527
528
528
529
529
530
531
532
532
533
533
534
534
535
535
536
536
537
537
538
538
539
539
540
541
542
542
543
543
544
544
545
545
546
546
547
547
548
548
549
549
550
551
552
552
553
553
554
554
555
555
556
556
557
557
558
558
559
559
560
561
562
562
563
563
564
564
565
565
566
566
567
567
568
568
569
569
570
571
572
572
573
573
574
574
575
575
576
576
577
577
578
578
579
579
580
581
582
582
583
583
584
584
585
585
586
586
587
587
588
588
589
589
590
591
592
592
593
593
594
594
595
595
596
596
597
597
598
598
599
599
600
601
602
602
603
603
604
604
605
605
606
606
607
607
608
608
609
609
610
611
612
612
613
613
614
614
615
615
616
616
617
617
618
618
619
619
620
620
621
621
622
622
623
623
624
624
625
625
626
626
627
627
628
628
629
629
630
630
631
631
632
632
633
633
634
634
635
635
636
636
637
637
638
638
639
639
640
640
641
641
642
642
643
643
644
644
645
645
646
646
647
647
648
648
649
649
650
650
651
651
652
652
653
653
654
654
655
655
656
656
657
657
658
658
659
659
660
660
661
661
662
662
663
663
664
664
665
665
666
666
667
667
668
668
669
669
670
670
671
671
672
672
673
673
674
674
675
675
676
676
677
677
678
678
679
679
680
680
681
681
682
682
683
683
684
684
685
685
686
686
687
687
688
688
689
689
690
690
691
691
692
692
693
693
694
694
695
695
696
696
697
697
698
698
699
699
700
700
701
701
702
702
703
703
704
704
705
705
706
706
707
707
708
708
709
709
710
710
711
711
712
712
713
713
714
714
715
715
716
716
717
717
718
718
719
719
720
720
721
721
722
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
730
731
731
732
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
740
741
741
742
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
750
751
751
752
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
760
761
761
762
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
```

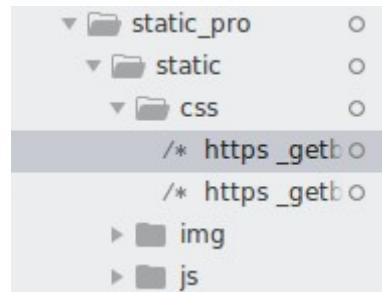
```

13
14
15 <!-- Bootstrap core CSS -->
16 <link href="/docs/5.0/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-e0JMYsd53ii+sc0/bJGFsicCzC+5NDVN2yr8+0RDqr0l0h+rP48ckxlpbzKgwra6" crossorigin="anonymous">
17
18 </!-- Favicons -->
19 <link rel="apple-touch-icon" href="/docs/5.0/assets/img/favicons/apple-touch-icon.png" sizes="180x180">
20 <link rel="icon" href="/docs/5.0/assets/img/favicons/favicon-32x32.png" sizes="32x32" type="image/png">
21 <link rel="icon" href="/docs/5.0/assets/img/favicons/favicon-16x16.png" sizes="16x16" type="image/png">
```

Abrimos los enlaces, hacemos click derecho y le damos a ‘Guardar como...’ y creamos una carpeta en la carpeta de ‘static’ dentro de nuestro proyecto llamada ‘css’ y ahí guardamos cada uno de los archivos:



Una vez guardados veremos que están en su respectiva carpeta. De paso, podemos crear las carpetas para las imágenes y los recursos JavaScript:



Para aplicar las hojas de estilo a la plantilla debemos añadir a nuestro ‘*base.html*’ unas líneas que ignoramos anteriormente porque no estábamos trabajando con la plantilla. Tenemos que ajustar los enlaces a como tenemos dispuestos los nuestros:

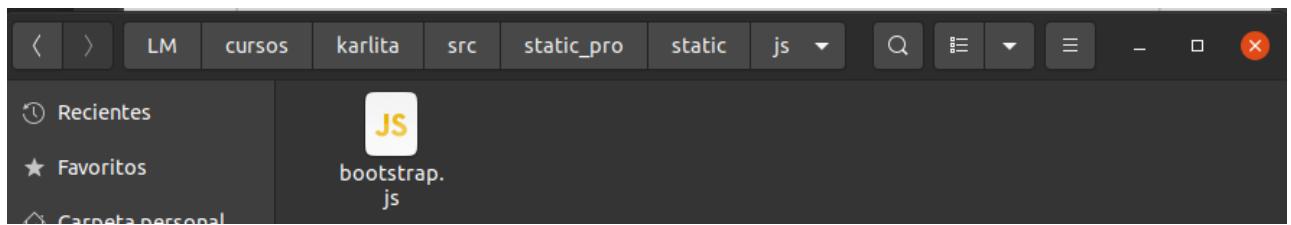
```
<link href="{% static '/home/alunsiito/Documentos/LM/cursos/karlita/src/static_pro/static/css/https_getbootstrap.com_docs_5.0_dist_css_bootstrap.min.css' %}" integrity="sha384-e03MYsd531i+sc0/b3GfscCz+9NDN2yrl+0RDqr8Ql0h+rP48cxlpbzKgwra0" crossorigin="anonymous">
<!-- Favicons -->
<link rel="apple-touch-icon" href="/docs/5.0/assets/img/favicons/apple-touch-icon.png" sizes="180x180">
<link rel="icon" href="/docs/5.0/assets/img/favicons/favicon-32x32.png" sizes="32x32" type="image/png">
<link rel="icon" href="/docs/5.0/assets/img/favicons/favicon-16x16.png" sizes="16x16" type="image/png">
<link rel="manifest" href="/docs/5.0/assets/img/favicons/manifest.json">
<link rel="mask-icon" href="/docs/5.0/assets/img/favicons/safari-pinned-tab.svg" color="#7952b3">
<link rel="icon" href="/docs/5.0/assets/img/favicons/favicon.ico">
<meta name="theme-color" content="#7952b3">

<style>
.bd-placeholder-img {
  font-size: 1.125rem;
  text-anchor: middle;
  -webkit-user-select: none;
  -moz-user-select: none;
  user-select: none;
}

@media (min-width: 768px) {
  .bd-placeholder-img-lg {
    font-size: 3.5rem;
  }
}
</style>

<!-- Custom styles for this template -->
<link href="{% static '/home/alunsiito/Documentos/LM/cursos/karlita/src/static_pro/static/css/https_getbootstrap.com_docs_5.0_examples_navbar-static-navbar-top.css' %}" rel="stylesheet">
</head>
</body>
```

Además, en mi caso al tener una plantilla distinta, he tenido que descargar un archivo JavaScript porque me saltaba el siguiente siguiente error en el terminal:



```
[16/Apr/2021 14:03:56] "GET / HTTP/1.1" 200 3498
Not Found: /docs/5.0/dist/js/bootstrap.bundle.min.js
[16/Apr/2021 14:03:56] "GET /docs/5.0/dist/js/bootstrap.bundle.min.js HTTP/1.1"
404 2660
```

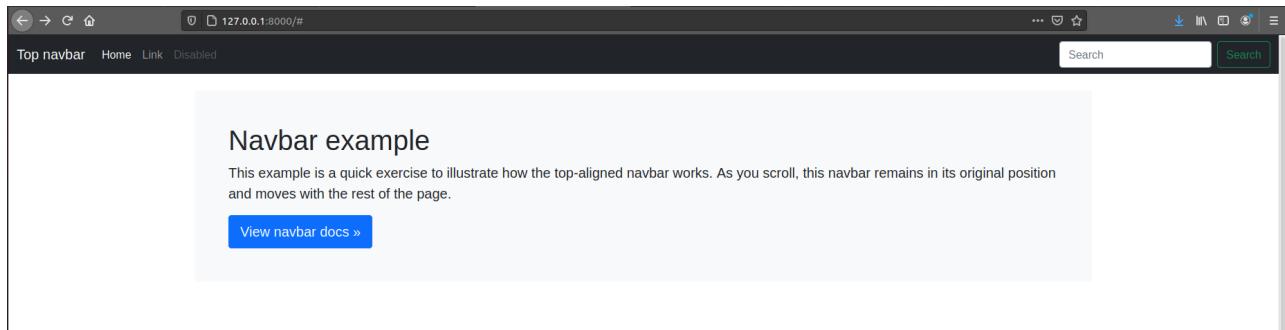
El mensaje me dice que no localiza ese archivo JavaScript, esto es lo que me he llevado a descargarlo. He cambiado los nombres de los archivos para que sea mas cómodo trabajar con ellos y quedaría tal que así:

```
10
11      <!-- Bootstrap core CSS -->
12      <link href="{% static 'css/bootstrap.min.css' %}" rel="stylesheet">
13
14
15
16
17
18
19
```

```
46
47      <!-- Custom styles for this template -->
48      <link href="{% static 'css/custom.css' %}" rel="stylesheet">
49
50
```

```
<script src="{% static 'js/bootstrap.js' %}" integrity="sha384-JEW9xMcG8R+pH31jmWH6WWPOWintQrMb4s7Z0dauHnUtxwoG2vI5DkLtS3qm9Ekf" crossorigin="anonymous"></script>
```

Y el resultado es el siguiente:



Subimos los nuevos archivos static..

```
(venv) alunsiito@alunsiito:~/Documentos/LM/cursos/karlita/src$ python3 manage.py collectstatic
You have requested to collect static files at the destination location as specified in your settings:
/home/alunsiito/Documentos/LM/cursos/karlita/static_env/static_root
This will overwrite existing files!
Are you sure you want to do this?
Type 'yes' to continue, or 'no' to cancel: yes
3 static files copied to '/home/alunsiito/Documentos/LM/cursos/karlita/static_env/static_root', 128 unmodified.
```

Y listo!

24. PLANTILLAS: HERENCIA, INCLUDE TAGS, BLOCKS

A continuación vamos a tratar con configuraciones de plantillas que van a hacer que ahorremos tiempo. Estas plantillas se renderizan en las vistas junto a la petición y contexto que almacenan. Vamos a coger como ejemplo las barras de navegación de la anterior plantilla para usarla para otros casos.

```
<nav class="navbar navbar-expand-md navbar-dark bg-dark mb-4">
<div class="container-fluid">
  <a class="navbar-brand" href="#">Top navbar</a>
  <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarCollapse" aria-controls="navbarCollapse" aria-expanded="false" aria-label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
  </button>
  <div class="collapse navbar-collapse" id="navbarCollapse">
    <ul class="navbar-nav me-auto mb-2 mb-md-0">
      <li class="nav-item">
        <a class="nav-link active" aria-current="page" href="#">Home</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#">Link</a>
      </li>
      <li class="nav-item">
        <a class="nav-link disabled" href="#" tabindex="-1" aria-disabled="true">Disabled</a>
      </li>
    </ul>
    <form class="d-flex">
      <input class="form-control me-2" type="search" placeholder="Search" aria-label="Search">
      <button class="btn btn-outline-success" type="submit">Search</button>
    </form>
  </div>
</div>
</nav>
```

En esto se basa el concepto de Herencia, el cual nos permite usar ciertos recursos para otras funciones.

Vamos a crear un nuevo archivo en ‘templates’ llamado ‘navbar.html’ y vamos a añadirle toda la barra de navegación que teníamos en el archivo base al nuevo:

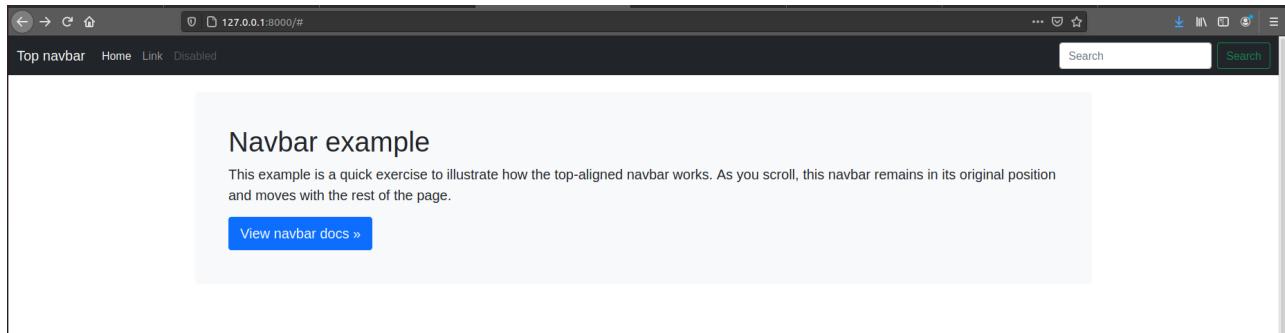
```
<nav class="navbar navbar-expand-md navbar-dark bg-dark mb-4">
<div class="container-fluid">
  <a class="navbar-brand" href="#">Top navbar</a>
  <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarCollapse" aria-controls="navbarCollapse" aria-expanded="false" aria-label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
  </button>
  <div class="collapse navbar-collapse" id="navbarCollapse">
    <ul class="navbar-nav me-auto mb-2 mb-md-0">
      <li class="nav-item">
        <a class="nav-link active" aria-current="page" href="#">Home</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#">Link</a>
      </li>
      <li class="nav-item">
        <a class="nav-link disabled" href="#" tabindex="-1" aria-disabled="true">Disabled</a>
      </li>
    </ul>
    <form class="d-flex">
      <input class="form-control me-2" type="search" placeholder="Search" aria-label="Search">
      <button class="btn btn-outline-success" type="submit">Search</button>
    </form>
  </div>
</div>
</nav>
```

Y donde está el archivo base tendremos que añadir los siguientes:

```
50  <body>
51
52  {% include "navbar.html" %}
53
54  <main class="container">
```

Guardamos y comprobamos el navegador.

La plantilla está igual que al principio:



Si cambiamos el título vemos como se cambia:

```

1  <nav class="navbar navbar-expand-md navbar-dark bg-dark mb-4">
2    <div class="container-fluid">
3      <a class="navbar-brand" href="#">PRUEBA DJANGO NAVBAR</a>
4      <button class="navbar-toggler" type="button" data-bs-toggle="collapse"
       data-bs-target="#navbarCollapse" aria-controls="navbarCollapse" aria-expanded="false" aria-label="Toggle navigation">
5        <span class="navbar-toggler-icon"></span>
6      </button>

```

A continuación, vamos a mover el contenedor que tiene todo el texto central al archivo de inicio...

```

1  <main class="container">
2    <div class="bg-light p-5 rounded">
3      <h1>Navbar example</h1>
4      <p class="lead">This example is a quick exercise to illustrate how the
       top-aligned navbar works. As you scroll, this navbar remains in its original
       position and moves with the rest of the page.</p>
5      <a class="btn btn-lg btn-primary" href="/docs/5.0/components/navbar/" role="button"
       >View navbar docs &raquo;</a>
6    </div>
7  </main>

```

Y donde estaba vamos a crear unos bloques de contenido el cual si hay que cerrar a diferencia del include:

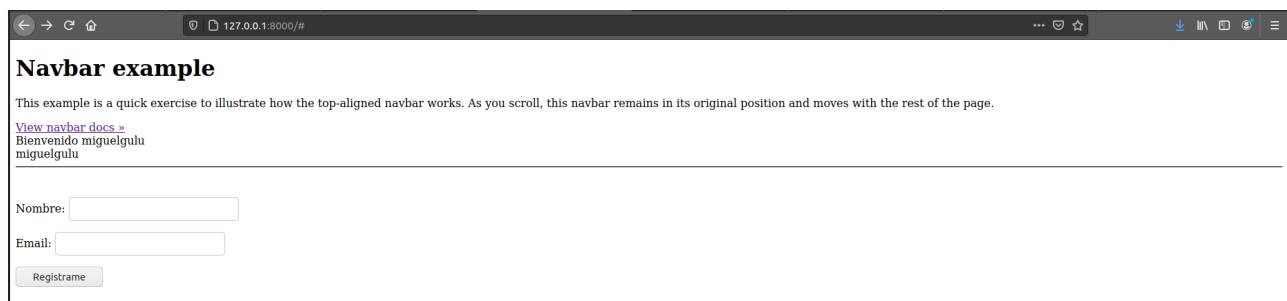
```
51
52     {% include "navbar.html" %}
53
54     {% block content %}
55         {% endblock %}
56
```

(Falta el '%' al final de block content)

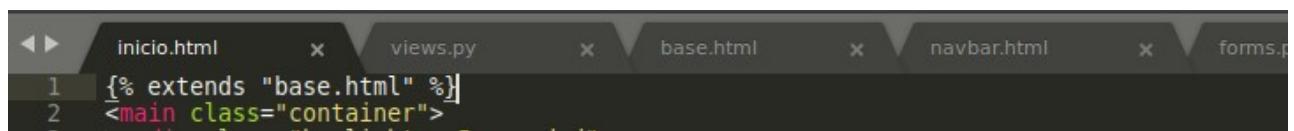
Guardamos, y nos vamos a 'views.py' y volvemos a cambiar el renderizado a 'inicio.html'.

```
38         #obj = Registrado.objects.create(email=abc, nombre=abc2)
39
40     return render_(request, "inicio.html", context)
41
42 def contact(reuest):
```

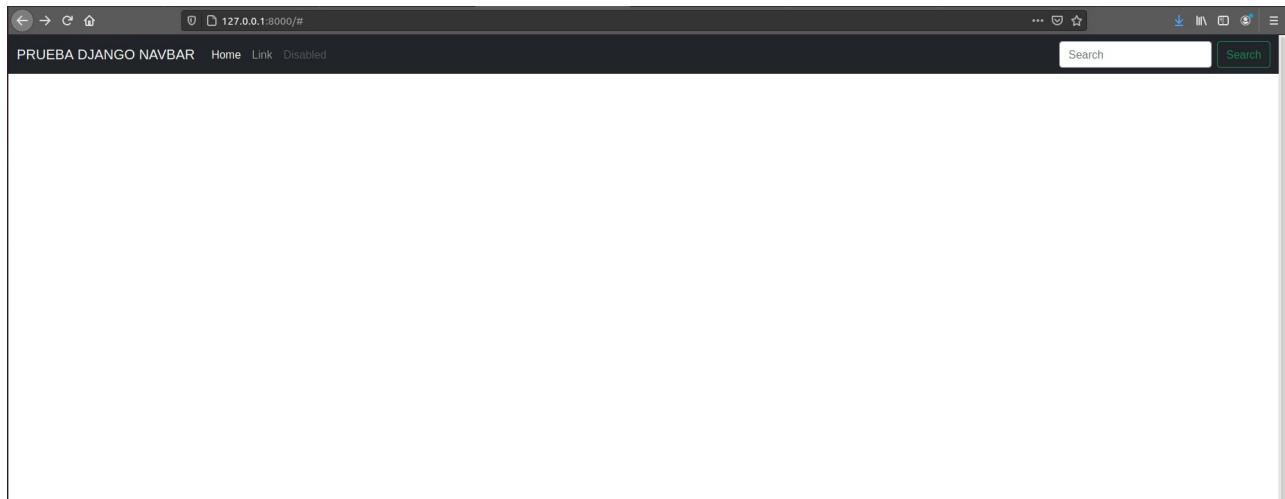
Guardamos y recargamos la página.



Vemos que no se ha añadido todo porque no hemos añadido un *extense* o una extensión que incluya 'base.html'...



Recargamos la página y veremos que nos falta el contenedor principal:



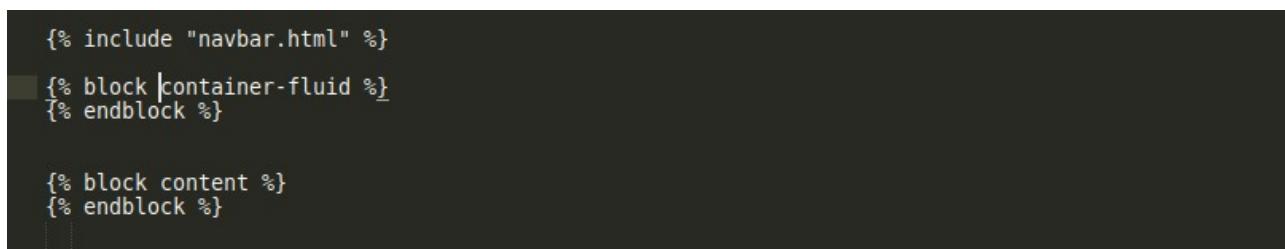
Si añadimos los tags de bloque pasará esto:

```
{% block content %}
<form method="POST" action="">{% csrf_token %}
{{ el_form.as_p }}
<input type="submit" value="Registrate" />
</form>
{% endblock %}
```



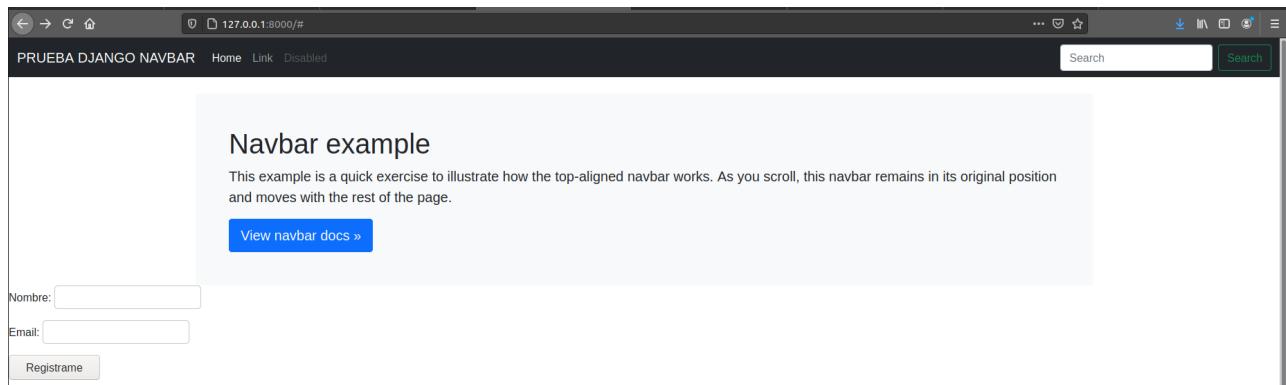
Saldrá nuestra página de inicio pero con el formulario que teníamos registrado al principio ya que todo lo que tenemos dentro del bloque es el propio formulario que creamos al principio. Al renderizar la vista el archivo va parte por parte hasta que llega aquí y añade este bloque.

Añadimos un nuevo bloque al ‘base.html’ el cual contendrá la barra de navegación.



Lo hacemos igual con ‘inicio.html’...

Guardamos, recargamos y vemos que se nos incorporá el contenedor con éxito:



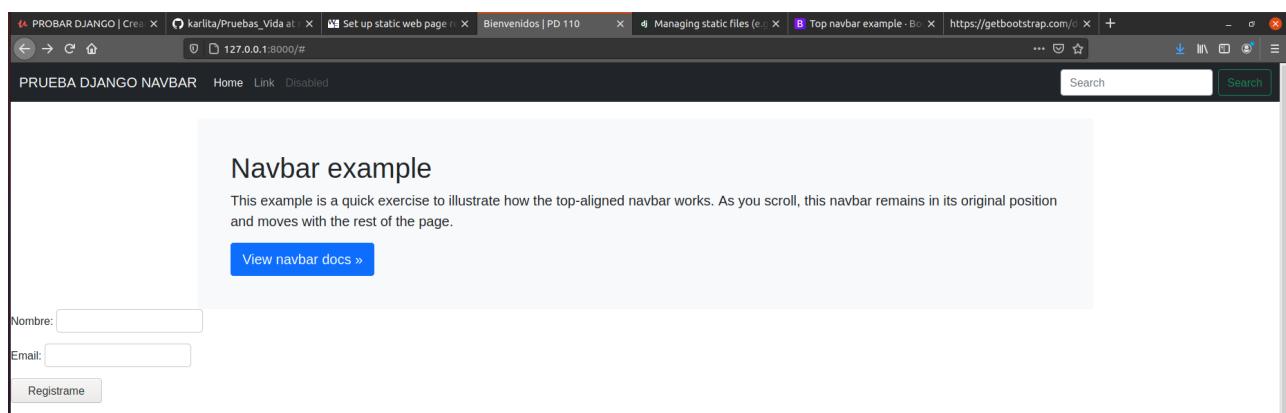
Otra cosa que podemos hacer es ir a '*base.html*' y en el título de la barra de navegación podemos añadir otro bloque para el título:

```
<meta name="generator" content="Hugo 0.82.0">
<title>{% block head_title %}{% endblock %}Top navbar example · Bootstrap v5.0</title>
```

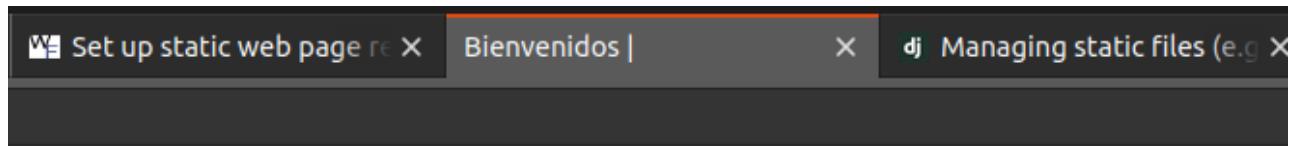
Podemos cambiar el título al de nuestro proyecto; copiamos lo anterior y lo pegamos en el inicio:

```
<% extends "base.html" %>
<% block head_title %>Bienvenidos | <% endblock %>
<% block container-fluid %>
<main class="container">
```

Y entonces el nombre de nuestra pestaña cambiará:



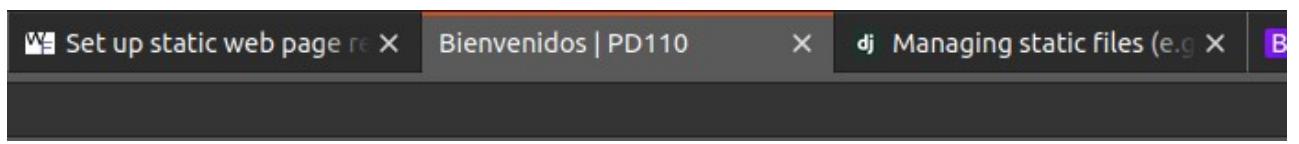
Si PD110 lo escribimos entre el inicio y el final del bloque saldrá lo siguiente:



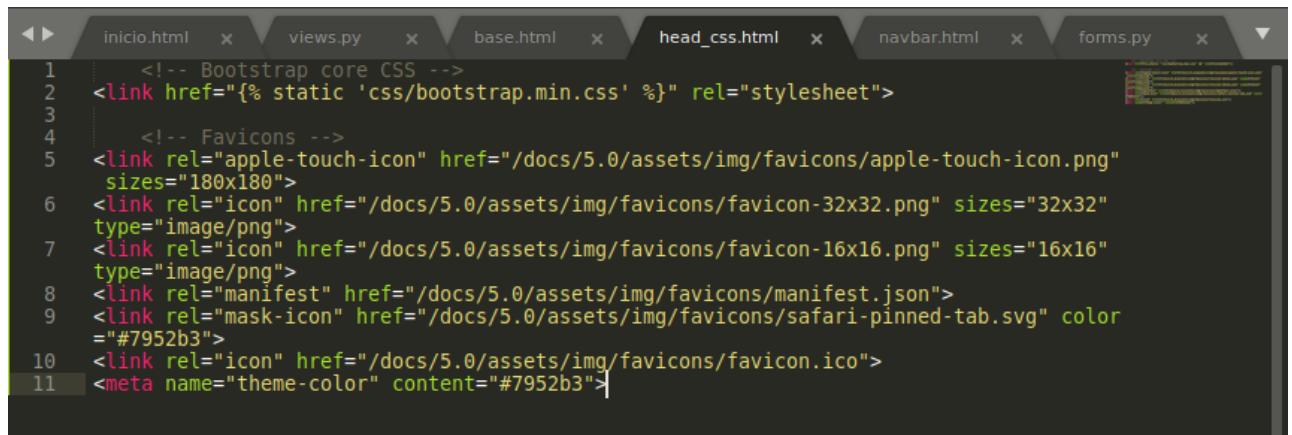
Sólo saldrá el bienvenido. Si queremos que salga el nombre del proyecto se puede incluir en inicio tal que así:

```

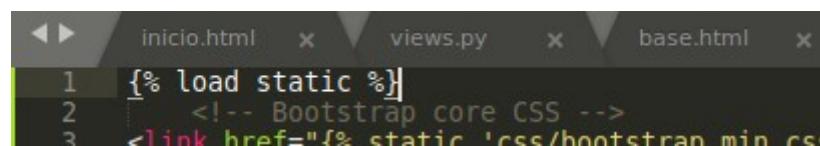
1  {% extends "base.html" %} 
2
3  {% block head_title %}Bienvenidos | {{ block.super }}{% endblock %} 
4
5  {% block container-fluid %} 
```



Vamos a seguir limpiando contenido del 'base.html'. Cortamos todo lo que significa el Bootstrap y lo pegamos en un nuevo template que lo llamaremos 'head_css.html':



Añadimos el 'load static'...



Y hacemos el *include* en el html base:

```

11     <title>{% block head_title %}FDIIV{% endblock %}</title>
12
13     <link rel="canonical" href="https://getbootstrap.com/docs/5.0/examples/
14         navbar-static/">
15
16     {% include "head_css.html" %}
17
18     <style>

```

Y la vista se mantendría exactamente igual, pero ahorraremos código en el html base.

También podemos crear bloques dentro de bloques:

```

39     {% include "navbar.html" %}
40
41
42     <main class="container">
43     {% block container-fluid %}
44         <div class="bg-light p-5 rounded">
45             {% block container-content %}
46
47             {% endblock %}
48         </div>
49     {% endblock %}
50
51     </main>

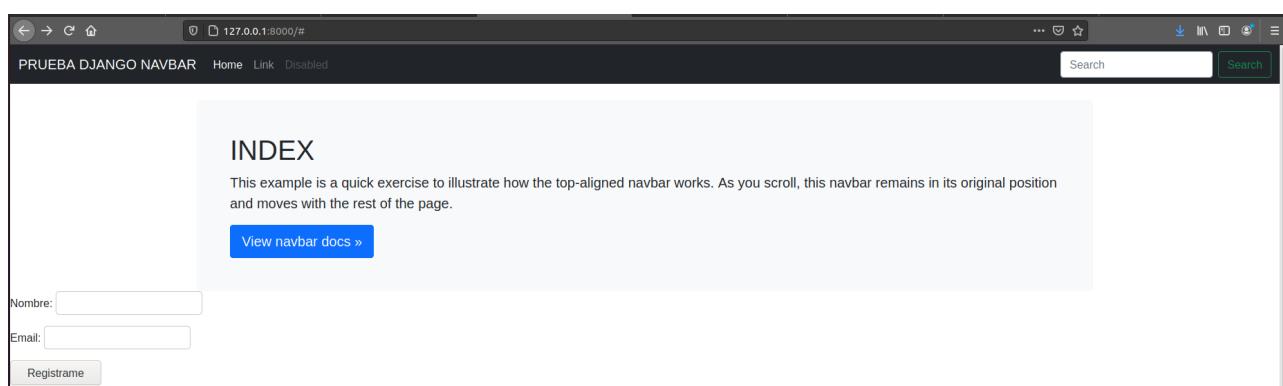
```

```

4
5     {% block container-content %}
6         <h1>INDEX</h1>
7             <p class="lead">This example is a quick exercise to illustrate how the
8                 top-aligned navbar works. As you scroll, this navbar remains in its original
9                 position and moves with the rest of the page.</p>
10            <a class="btn btn-lg btn-primary" href="/docs/5.0/components/navbar/" role="button"
11                ">View navbar docs &raquo;</a>
12
13     {% endblock %}
14     {{ titulo }}<br/>

```

Y el resultado es el siguiente:

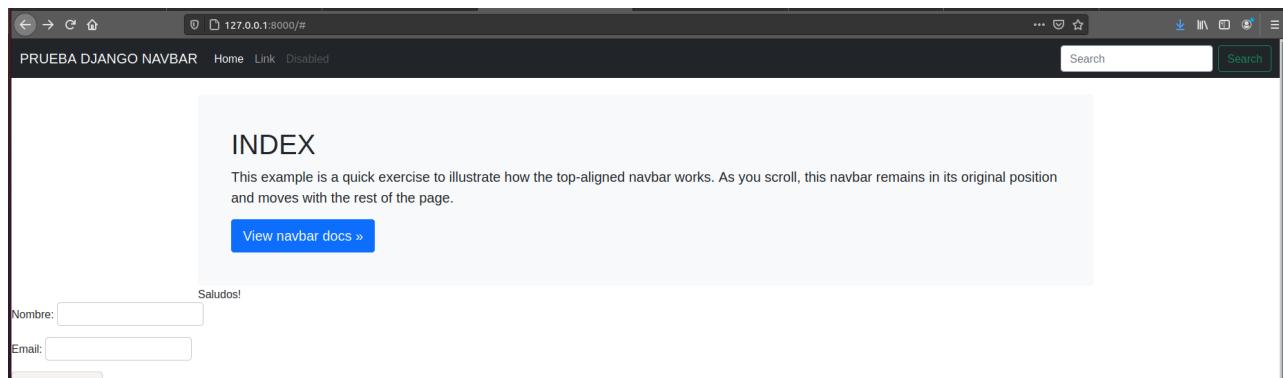


Es exactamente lo mismo pero con el título del *container* cambiado.

Todo esto se hace en base al curso pero, al ser una plantilla cambiada, hay que ajustarlo al curso.

```

9  {% endblock %}
10
11  {% block container-fluid %}
12  {{ block.super }}
13  Saludos!
14  {% endblock %}
15
16  {{ titulo }}<br/>
17  {{ request.user }}<br/>
18  <hr/>
```



Esta última opción la vamos a quitar, ya que es una mera demostración.

25. DJANGO CRISPY FORMS

Crispy es una aplicación de Django que permite fácilmente añadir, customizar y reusar formularios para darle un mayor estilo a nuestra página.

La página como la dejamos en el ejercicio anterior es muy simple y como poco elementos visuales que destaque. Para instalar *crispy-forms* tenemos que ir a su respectiva documentación e instalar los recursos mediante pip3:

```
(venv) alunsiito@alunsiito:~/Documentos/LM/cursos/karlita/src$ pip3 install --upgrade django-crispy-forms
```

```
(venv) alunsiito@alunsiito:~/Documentos/LM/cursos/karlita/src$ pip3 install --upgrade django-crispy-forms
Collecting django-crispy-forms
  Using cached django_crispy_forms-1.11.2-py3-none-any.whl (112 kB)
Installing collected packages: django-crispy-forms
Successfully installed django-crispy-forms-1.11.2
(venv) alunsiito@alunsiito:~/Documentos/LM/cursos/karlita/src$
```

Ahora debemos añadirlo a nuestras **INSTALLED_APPS**:

```

38 INSTALLED_APPS = [
39     #Django
40     'django.contrib.admin',
41     'django.contrib.auth',
42     'django.contrib.contenttypes',
43     'django.contrib.sessions',
44     'django.contrib.messages',
45     'django.contrib.staticfiles',
46     #Terceros
47     'boletin.apps.BoletinConfig',
48     'crispy_forms',
49 ]
50

```

Y hacemos las respectivas migraciones para guardar todo en la base de datos:

```

(vENV) alunsiito@alunsiito:~/Documentos/LM/cursos/karlita/src$ python3 manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, boletin, contenttypes, sessions
Running migrations:
  No migrations to apply.
(vENV) alunsiito@alunsiito:~/Documentos/LM/cursos/karlita/src$ █

```

Y tenemos que establecer el pack predeterminado de *templates*:

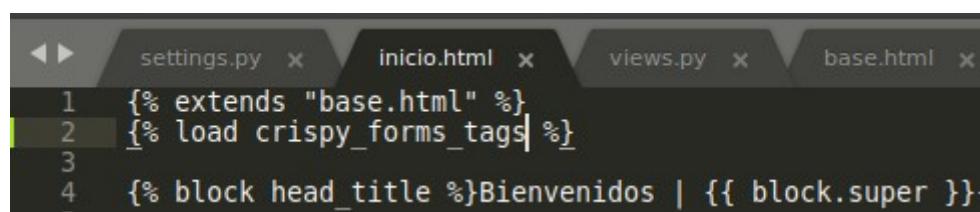
```

61 ROOT_URLCONF = 'pd110.urls'
62
63 CRISPY_TEMPLATE_PACK = 'bootstrap4'
64
65 TEMPLATES = [
66

```

Como usamos *bootstrap* para nuestra aplicación, es lo que ha que definir en la variable (última versión, karlita usa la 3).

Para usar *crispy-forms* debemos añadirla al principio de nuestro archivo html *inicio*:



```

settings.py × inicio.html × views.py × base.html ×
1  {% extends "base.html" %}
2  {% load crispy_forms_tags %}
3
4  {% block head_title %}Bienvenidos | {{ block.super }}{% endblock %}

```

Y cambiamos el formato del formulario de ‘as_p’ a ‘crispy’:

```
18  {% block content %}  
19  <form method="POST" action="">{% csrf_token %}  
20  {{ el_form|crispy }}  
21  <input type="submit" value="Registrate" />  
22  </form>  
23  {% endblock %}
```

Y como podemos comprobar, el formulario ha cambiado su diseño.

The screenshot shows a web browser window with the URL 127.0.0.1:8000. The page title is 'PRUEBA DJANGO NAVBAR'. The main content area has a heading 'INDEX' and a paragraph: 'This example is a quick exercise to illustrate how the top-aligned navbar works. As you scroll, this navbar remains in its original position and moves with the rest of the page.' Below this is a 'View navbar docs »' button. The form itself consists of three fields: 'Nombre' (text input), 'Email*' (text input with a red border), and 'Registrate' (blue submit button). The entire form is styled using the crispy_forms library.

This screenshot shows the same registration form as the previous one, but with all three fields ('Nombre', 'Email*', and 'Registrate') highlighted with a red border, suggesting they are required or have validation errors.

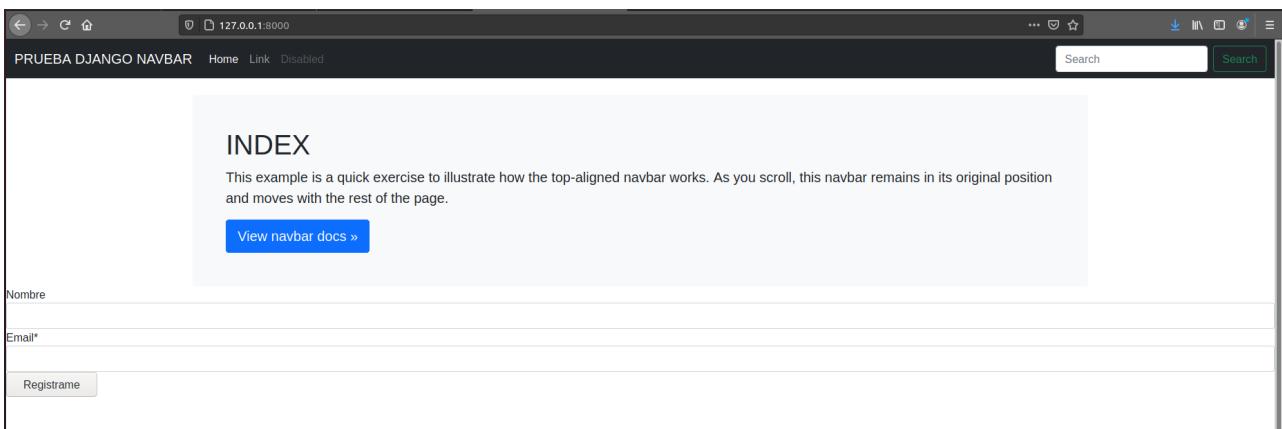
26. ESTILO: BOOTSTRAP 1

Si quisieramos cambiar el formato de nuestro contenedor de Bootstrap bastaría con añadir fluid en la siguiente línea (esto ya viene en el nuestro porque en nuestra plantilla lo tiene por defecto):

```
1  <nav class="navbar navbar-expand-md navbar-dark bg-dark mb-4">
2    <div class="container-fluid">
3      <a class="navbar-brand" href="#">PRUEBA DJANGO NAVBAR</a>
```

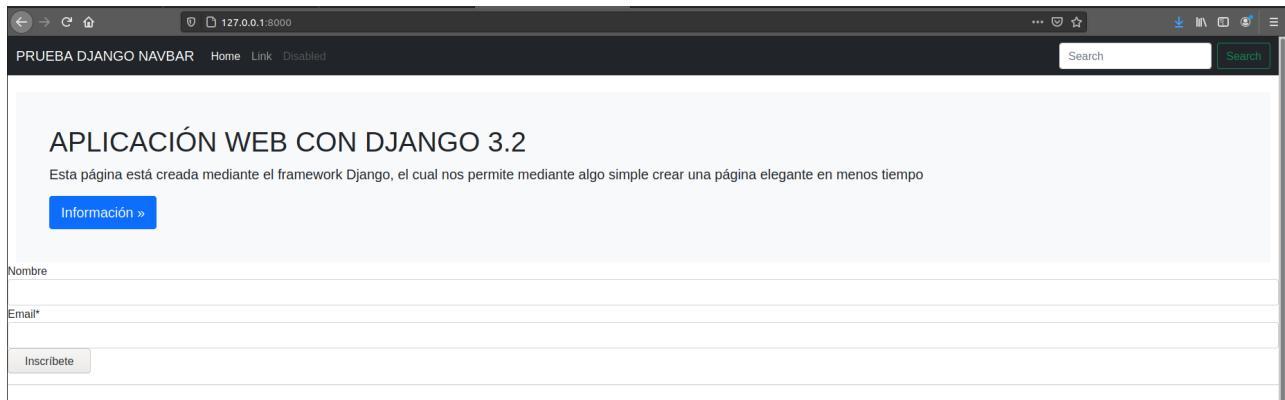
```
10
11
12  <main class="container-fluid">
13    {% block container-fluid %}
```

Con esta opción los contenedores van ajustados a la página:



Ahora vamos a cambiar un poco el contenido del contenedor principal:

```
1  {% extends "base.html" %}
2  {% load crispy_forms_tags %}
3
4  {% block head_title %}Bienvenidos | {{ block.super }}{% endblock %}
5
6  {% block container-content %}
7    <h1>APLICACION WEB CON DJANGO 3.2</h1>
8    <p>Esta página está creada mediante el framework Django, el cual nos permite mediante algo simple crear una página elegante en menos tiempo</p>
9    <a href="/docs/5.0/components/navbar/" role="button">Información</a>
10   {% endblock %}
11
12
13  {{ titulo }}<br/>
14  {{ request.user }}<br/>
15  <hr/>
16  <hr/>
17
18  {% block content %}
19  <form method="POST" action="{% csrf_token %}>
20  {{ el_form|crispy }}>
21  <input type="submit" value="Inscribete" />
22  </form>
23  <hr/>
24  {% endblock %}
```



Vamos a darle un poco de espaciado a nuestra página:

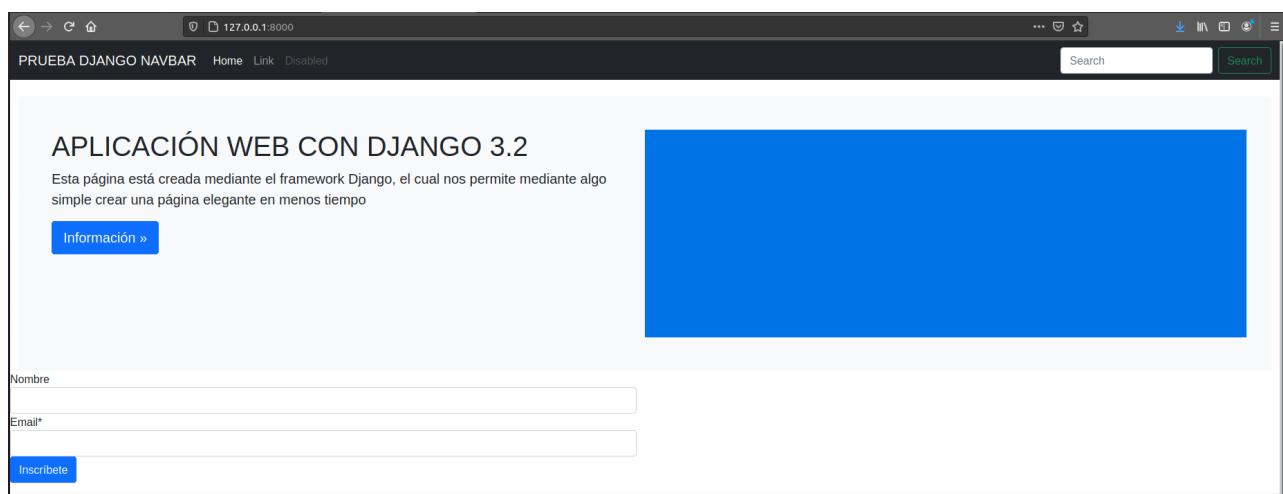
```
7  <div class='row'>
8      <div class='col-sm-6'>
9          <h1>APLICACIÓN WEB CON DJANGO 3.2</h1>
10         <p class="lead">Esta página está creada mediante el framework Django, el cual
11             nos permite mediante algo simple crear una página elegante en menos tiempo</p>
12         <a class="btn btn-lg btn-primary" href="/docs/5.0/components/navbar/" role="button">Información &raquo;</a>
13     <% endblock %>
14 </div>
```



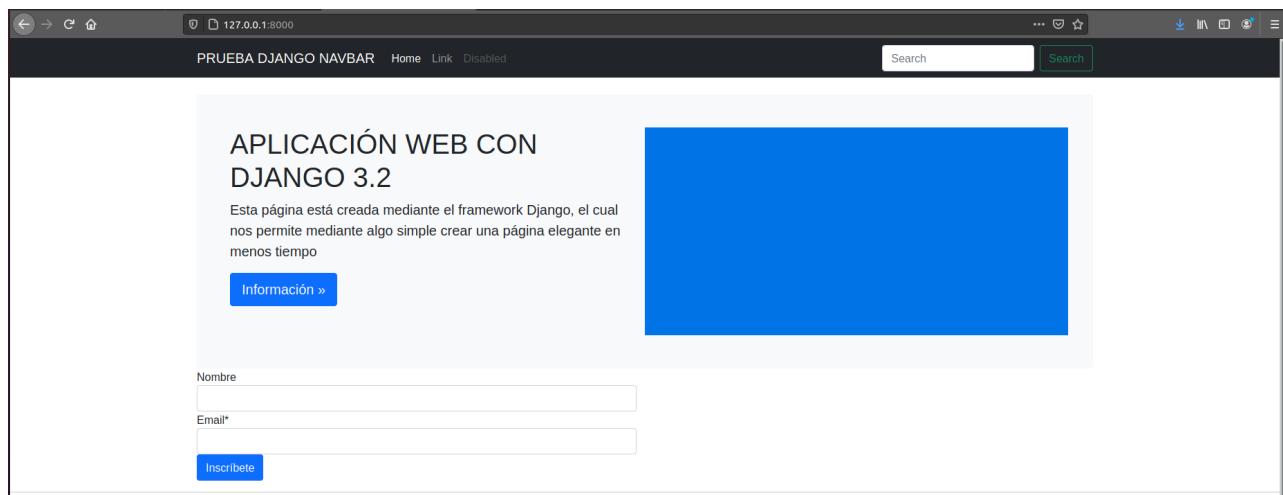
Vamos a añadir contenido en el hueco ese que creamos anteriormente:

```
7   <div class='row'>
8     <div class='col-sm-6'>
9       <h1>APLICACIÓN WEB CON DJANGO 3.2</h1>
10      <p class="lead">Esta página está creada mediante el framework Django, el cual
11         nos permite mediante algo simple crear una página elegante en menos tiempo</p>
12      <a class="btn btn-lg btn-primary" href="/docs/5.0/components/navbar/" role="button">Información &raquo;</a>
13    </div>
14  <div class='col-sm-6' style='background-color:#0073e6; height:300px;'></div>
15 </div>
16  {% endblock %}
```

Y se vería algo así:



Luego tenemos la opción quitarle la opción de fluid al contenedor para que se vea un poco mejor visualmente:

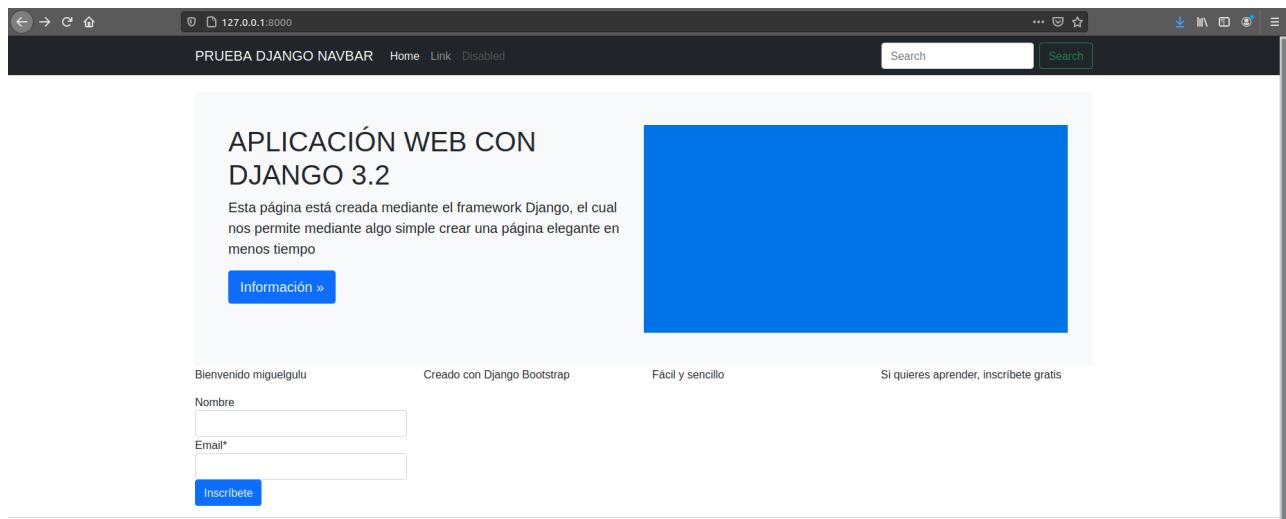


Vamos a seguir añadiendo columnas de contenido...

```

24 <div class="container">
25 <div class="row">
26   <div class="col-sm-3">
27     <p>{{ titulo }}</p>
28     <form method="POST" action="">{{ csrf_token %}
29       {{ el_form|crispy }}
30       <input class='btn btn-primary' type="submit" value="Inscríbete" />
31     </form>
32   </div>
33   <div class="col-sm-3">
34     <p>Creado con Django Bootstrap</p>
35   </div>
36   <div class="col-sm-3">
37     <p>Fácil y sencillo</p>
38   </div>
39   <div class="col-sm-3">
40     <p>Si quieres aprender, inscríbete gratis</p>
41   </div>
42   </div>
43 </div>
44
45 <hr/>
```

Y el resultado sería este:



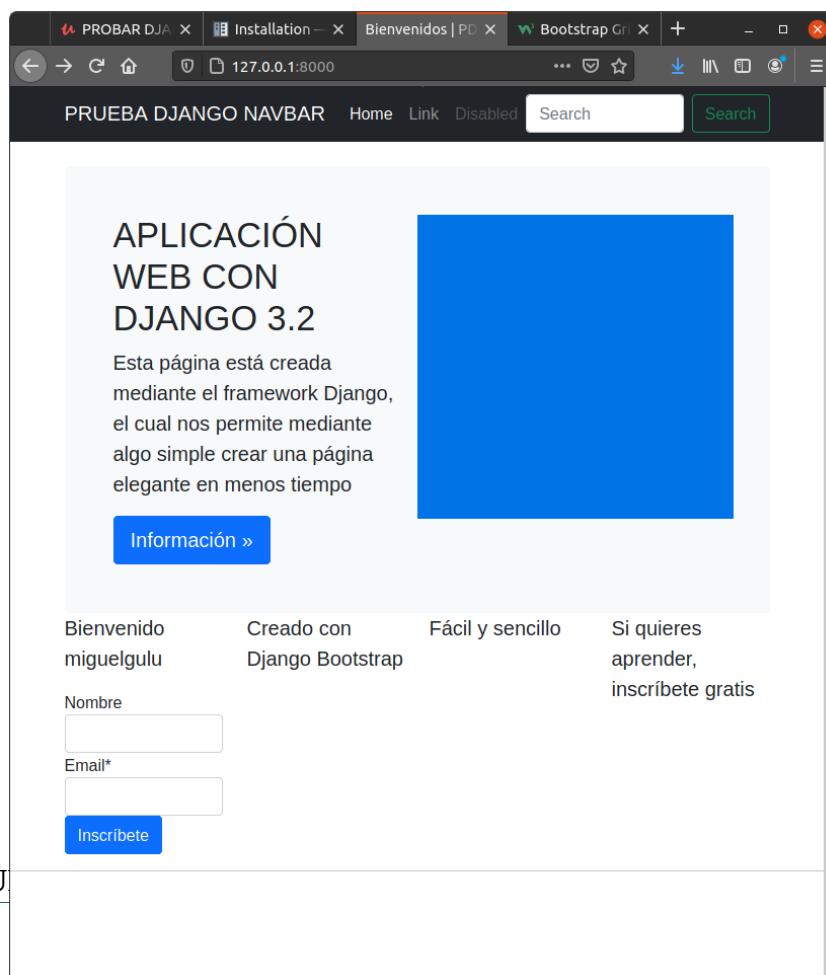
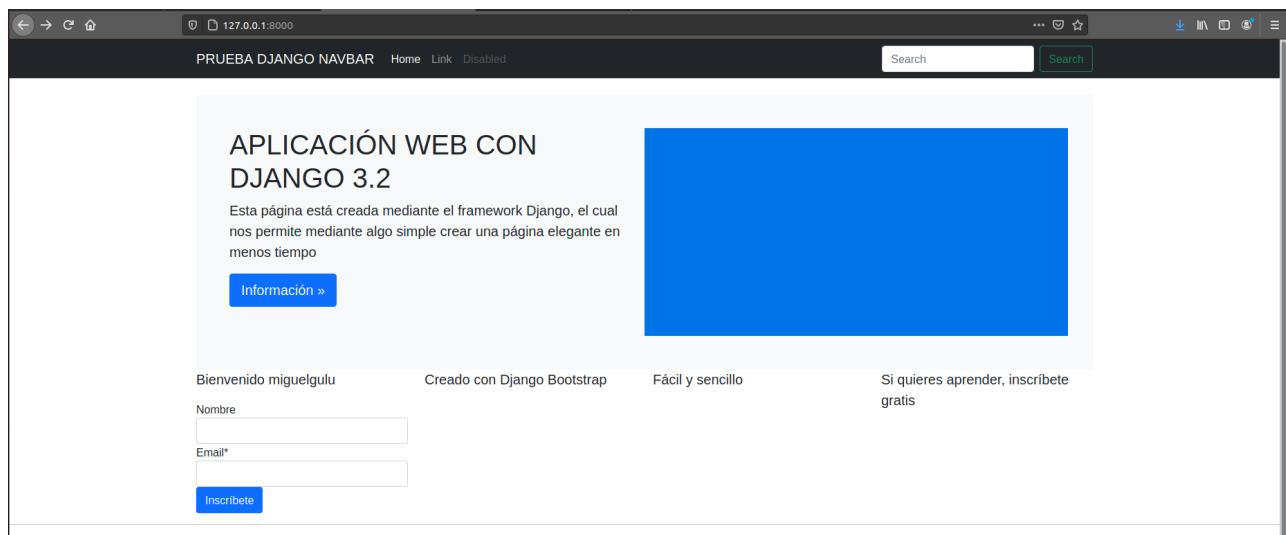
Karlita pone el registro a la derecha, pero yo prefiero dejarlo a la izquierda.

Finalmente vamos a añadirle algo más de tamaño a las letras con la clase '*lead*'.

```

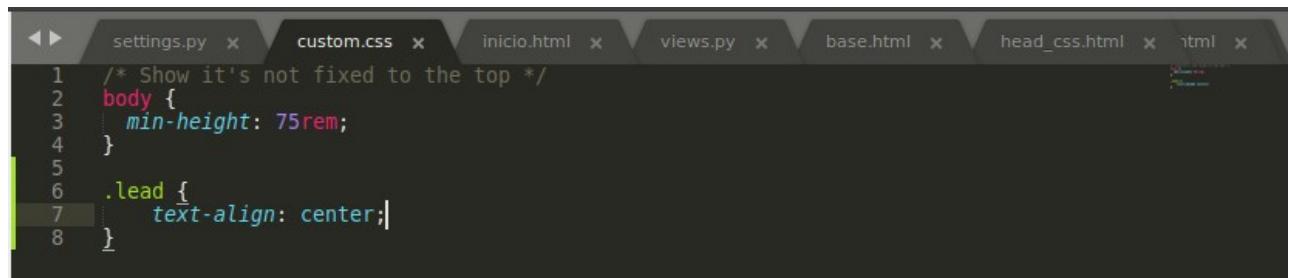
24 <div class="container">
25 <div class="row">
26   <div class="col-sm-3">
27     <p class="lead">{{ titulo }}</p>
28     <form method="POST" action="">{{ csrf_token %}
29       {{ el_form|crispy }}
30       <input class='btn btn-primary' type="submit" value="Inscríbete" />
31     </form>
32   </div>
33   <div class="col-sm-3">
34     <p class="lead">Creado con Django Bootstrap</p>
35   </div>
36   <div class="col-sm-3">
37     <p class="lead">Fácil y sencillo</p>
38   </div>
39   <div class="col-sm-3">
40     <p class="lead">Si quieres aprender, inscríbete gratis</p>
41   </div>
42   </div>
43 </div>
44
45 <hr/>
```

Y como se adapta al tamaño de la ventana:



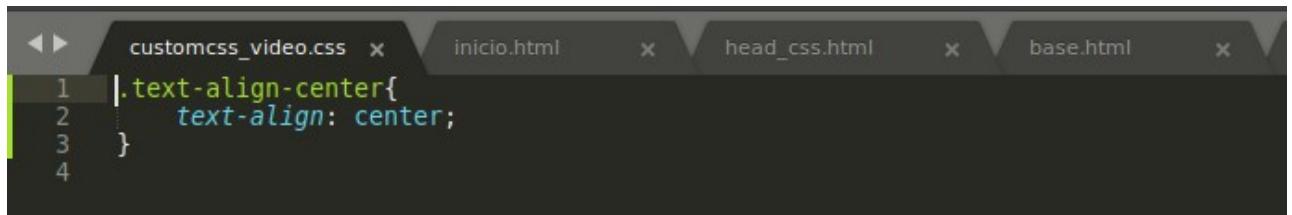
27. ESTILO: CSS CUSTOM

En esta práctica vamos a crear una nueva hoja de estilos para ajustar los títulos de nuestra página. Podríamos crear un nuevo archivo css, pero vamos a aprovechar el archivo CSS que ya tenemos de ‘custom.css’:



```
1 /* Show it's not fixed to the top */
2 body {
3     min-height: 75rem;
4 }
5
6 .lead {
7     text-align: center;
8 }
```

Esto puede dar lugar a muchos problemas, así que lo suyo es crear una nueva y crear una nueva clase:



```
1 .text-align-center{
2     text-align: center;
3 }
4
```

Efectivamente se ha registrado el nuevo archivo CSS.

Nos faltaría un *collect* para subirlo todo...

```
(venv) alunsiito@alunsiito:~/Documentos/LM/cursos/karlita/src$ python3 manage.py collectstatic
You have requested to collect static files at the destination location as specified in your settings:
/home/alunsiito/Documentos/LM/cursos/karlita/static_env/static_root
This will overwrite existing files!
Are you sure you want to do this?
Type 'yes' to continue, or 'no' to cancel: yes
2 static files copied to '/home/alunsiito/Documentos/LM/cursos/karlita/static_env/static_root', 130 unmodified.
(venv) alunsiito@alunsiito:~/Documentos/LM/cursos/karlita/src$
```

```
[18/Apr/2021 14:46:58] "GET /media/js/bootstrap.js HTTP/1.1" 304 0
[18/Apr/2021 14:46:58] "GET /media/css/customcss_video.css HTTP/1.1" 304 0
[18/Apr/2021 14:47:15] "GET / HTTP/1.1" 200 5033
[18/Apr/2021 14:47:15] "GET /media/js/bootstrap.js HTTP/1.1" 304 0
```

Y ya tenemos todos los archivos static subidos!

28. ENLACES CON NOMBRES URL

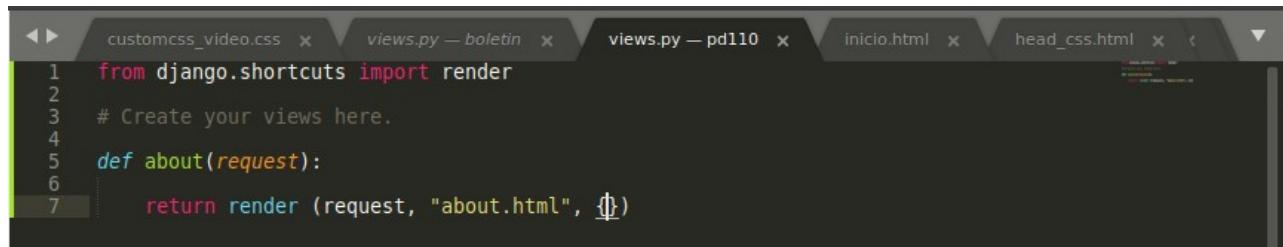
Nuestra página tiene una barra de navegación la cual si tuvieras configurada podríamos usarla para navegar a otras páginas o direcciones. Esta parte del curso va a hacer eso, configurarla para usarla a nuestro antojo.

Para ello, nos vamos a hacer un nuevo archivo de vistas pero dentro de nuestro proyecto y no de la aplicación en si. Lo creamos y le añadimos solamente lo siguiente:



```
customcss_video.css x views.py — boletin x views.py — pd110 x inicio.html x head_css.html x
1 from django.shortcuts import render
2
3 # Create your views here.
4
5 def inicio(request):
6     return render (request, "inicio.html", context)
```

La cual se quedará así:



```

1 from django.shortcuts import render
2
3 # Create your views here.
4
5 def about(request):
6     return render (request, "about.html", {})
7

```

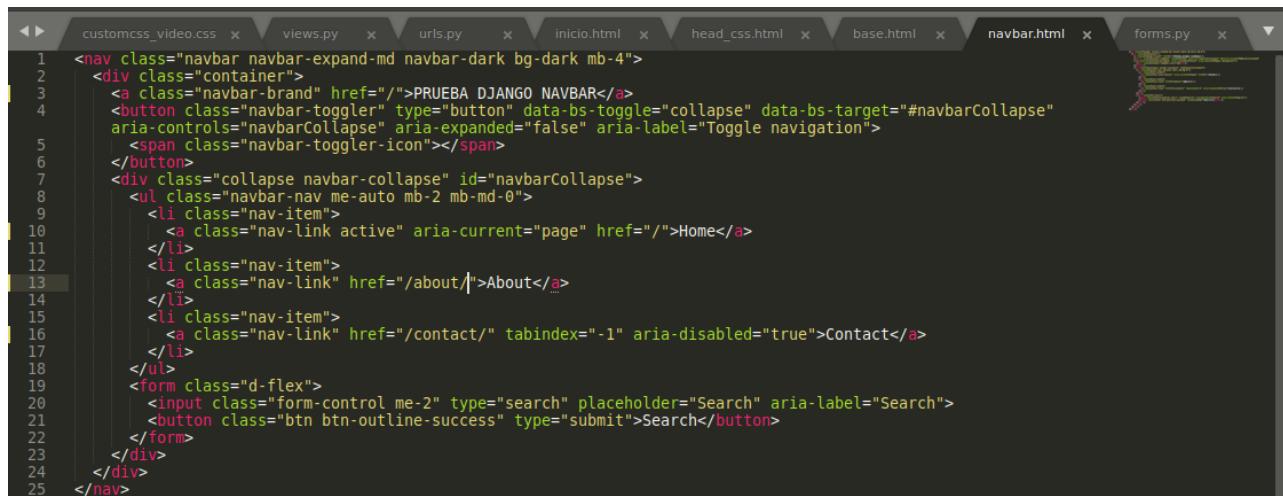
Y la tendremos que importar a nuestras URLs:

```

22 from boletin import views
23 from .views import about
24
25 urlpatterns = [
26     path('admin/', admin.site.urls),
27     path('', views.inicio, name='inicio'),
28     path('contact/', views.contact, name='contact')
29     path('about/', about, name='about')
30 ]
31

```

Ajustamos las URLs de nuestra plantilla y escribimos ‘/’ en vez de '#':

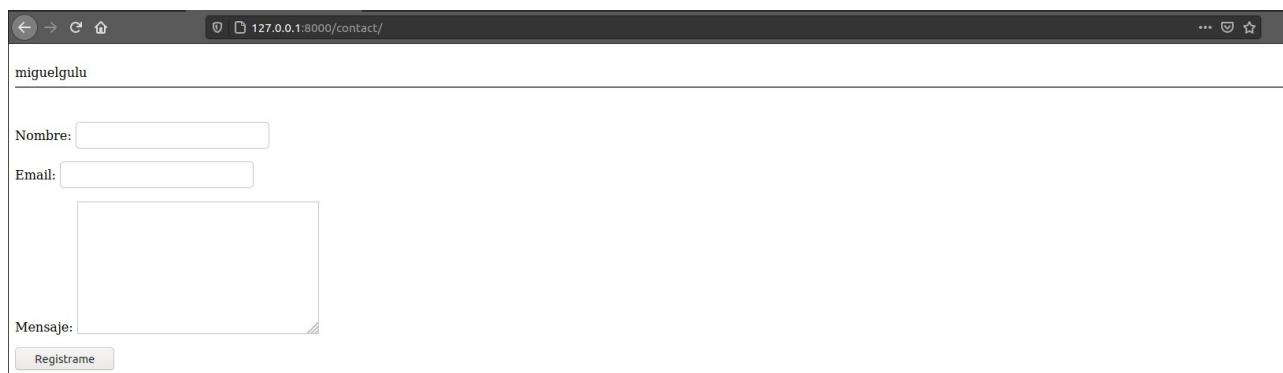


```

1 <nav class="navbar navbar-expand-md navbar-dark bg-dark mb-4">
2   <div class="container">
3     <a class="navbar-brand" href="/">PRUEBA DJANGO NAVBAR</a>
4     <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarCollapse"
5       aria-controls="navbarCollapse" aria-expanded="false" aria-label="Toggle navigation">
6       <span class="navbar-toggler-icon"></span>
7     </button>
8     <div class="collapse navbar-collapse" id="navbarCollapse">
9       <ul class="navbar-nav me-auto mb-2 mb-md-0">
10         <li class="nav-item">
11           <a class="nav-link active" aria-current="page" href="/">Home</a>
12         </li>
13         <li class="nav-item">
14           <a class="nav-link" href="/about/">About</a>
15         </li>
16         <li class="nav-item">
17           <a class="nav-link" href="/contact/" tabindex="-1" aria-disabled="true">Contact</a>
18         </li>
19       </ul>
20       <form class="d-flex">
21         <input class="form-control me-2" type="search" placeholder="Search" aria-label="Search">
22         <button class="btn btn-outline-success" type="submit">Search</button>
23       </form>
24     </div>
25   </nav>

```

Y si en nuestra plantilla hacemos click en ‘Contact’...



miguelgulu

Nombre:

Email:

Mensaje:

Y si lo hacemos en ‘About’...

TemplateDoesNotExist at /about/
about.html

Request Method: GET
Request URL: http://127.0.0.1:8000/about/
Django Version: 3.2
Exception Type: TemplateDoesNotExist
Exception Value: about.html
Exception Location: /home/alunusito/Dокументos/LM/cursos/karlita/venv/lib/python3.8/site-packages/django/template/loader.py, line 19, in get_template
Python Executable: /home/alunusito/Dокументos/LM/cursos/karlita/venv/bin/python3
Python Version: 3.8.5
Python Path: ['/home/alunusito/Dокументos/LM/cursos/karlita/src',
 '/usr/lib/python3.8',
 '/usr/lib/python3.8/lib-dynload',
 '/home/alunusito/Dокументos/LM/cursos/karlita/venv/lib/python3.8/site-packages']
Server time: Tue, 20 Apr 2021 14:10:51 +0000

Template-loader postmortem

Django tried loading these templates, in this order:

- Using engine django:
 - django.templatetags.loader_tags.Loader: /home/alunusito/Dокументos/LM/cursos/karlita/src/templates/about.html (Source does not exist)
 - django.template.loaders.app_directories.Loader: /home/alunusito/Dокументos/LM/cursos/karlita/venv/lib/python3.8/site-packages/django/contrib/admin/templates/about.html (Source does not exist)
 - django.template.loaders.app_directories.Loader: /home/alunusito/Dокументos/LM/cursos/karlita/venv/lib/python3.8/site-packages/django/contrib/auth/templates/about.html (Source does not exist)
 - django.template.loaders.app_directories.Loader: /home/alunusito/Dокументos/LM/cursos/karlita/venv/lib/python3.8/site-packages/crispy_forms/templates/about.html (Source does not exist)

Traceback [Switch to copy-and-paste view](#)

```
/home/alunusito/Dокументos/LM/cursos/karlita/venv/lib/python3.8/site-packages/django/core/handlers/exception.py, line 47, in inner
    47.     response = get_response(request)
▶ Local vars
/home/alunusito/Dокументos/LM/cursos/karlita/venv/lib/python3.8/site-packages/django/core/handlers/base.py, line 181, in _get_response
    181.         response = wrapped_callback(request, *callback_args, **callback_kwargs)
▶ Local vars
/home/alunusito/Dокументos/LM/cursos/karlita/src/pd110/views.py, line 6, in about
    6.     return render(request, "about.html", {})
▶ Local vars
/home/alunusito/Dокументos/LM/cursos/karlita/venv/lib/python3.8/site-packages/django.shortcuts.py, line 19, in render
    19.     content = loader.render_to_string(template_name, context, request, using=using)
▶ Local vars
/home/alunusito/Dокументos/LM/cursos/karlita/venv/lib/python3.8/site-packages/django/template/loader.py, line 61, in render_to_string
    61.     template = get_template(template_name, using=using)
```

Nos sale el error de que la plantilla no existe.

Ahora tenemos el problema de que si queremos cambiar la url en ‘urlpatterns’ tendríamos que cambiarlo en el navbar, por lo que la manera de solucionar esto es poner la dirección en el navbar como variable:

```
13.     <li><a class="nav-link" href="{% url 'about' %}>About</a>
14.   </li>
15.   <li class="nav-item">
16.     <a class="nav-link" href="{% url 'contact' %}" tabindex="-1" aria-disabled="true">Contact</a>
17.   </li>
```

Guardamos, recargamos y nos saldrá la página como al principio.

Si cambiamos la url desde el archivo de ‘url.py’ igualmente funcionará:

The screenshot shows a browser window with the URL `127.0.0.1:8000/jdasaldnsajlkdsdf/`. The page displays an error message: "TemplateDoesNotExist at /jdasaldnsajlkdsdf/". Below the title, there is detailed error information:

```
about.html
Request Method: GET
Request URL: http://127.0.0.1:8000/jdasaldnsajlkdsdf/
Django Version: 3.2
Exception Type: TemplateDoesNotExist
Exception Value: about.html
Exception Location: /home/alunusito/Documentos/LM/cursos/karilita/venv/lib/python3.8/site-packages/django/template/loader.py, line 19, in get_template
Python Executable: /home/alunusito/Documentos/LM/cursos/karilita/venv/bin/python3
Python Version: 3.8.5
Python Path: ['/home/alunusito/Documentos/LM/cursos/karilita/src',
 '/usr/lib/python3.zip',
 '/usr/lib/python3.8',
 '/usr/lib/python3.8/lib-dynload',
 '/home/alunusito/Documentos/LM/cursos/karilita/venv/lib/python3.8/site-packages']
Server time: Tue, 20 Apr 2021 14:19:51 +0000
```

Template-loader postmortem

Django tried loading these templates, in this order:

Using engine django:

- django.template.loaders.filesystem.Loader: /home/alunusito/Documentos/LM/cursos/karilita/src/templates/about.html (Source does not exist)
- django.template.loaders.app_directories.Loader: /home/alunusito/Documentos/LM/cursos/karilita/venv/lib/python3.8/site-packages/django/contrib/admin/templates/about.html (Source does not exist)
- django.template.loaders.app_directories.Loader: /home/alunusito/Documentos/LM/cursos/karilita/venv/lib/python3.8/site-packages/django/contrib/auth/templates/about.html (Source does not exist)
- django.template.loaders.app_directories.Loader: /home/alunusito/Documentos/LM/cursos/karilita/venv/lib/python3.8/site-packages/crispy_forms/templates/about.html (Source does not exist)

Traceback [Switch to copy-and-paste view](#)

```
/home/alunusito/Documentos/LM/cursos/karilita/venv/lib/python3.8/site-packages/django/core/handlers/exception.py, line 47, in inner
    response = get_response(request)
▶ Local vars
/home/alunusito/Documentos/LM/cursos/karilita/venv/lib/python3.8/site-packages/django/core/handlers/base.py, line 181, in _get_response
    response = wrapped_callback(request, *callback_args, **callback_kwargs)
```

29. ESTILO: BOOTSTRAP 2

Al igual que en la práctica del vídeo 26, vamos a modificar un poco las plantillas de ‘Bootstraps’ pero los vamos a hacer con el formulario de contacto. Nos vamos a nuestro ‘forms.html’ y añadimos la extensión de nuestro ‘base.html’ y nuestro ‘crispy_forms’:

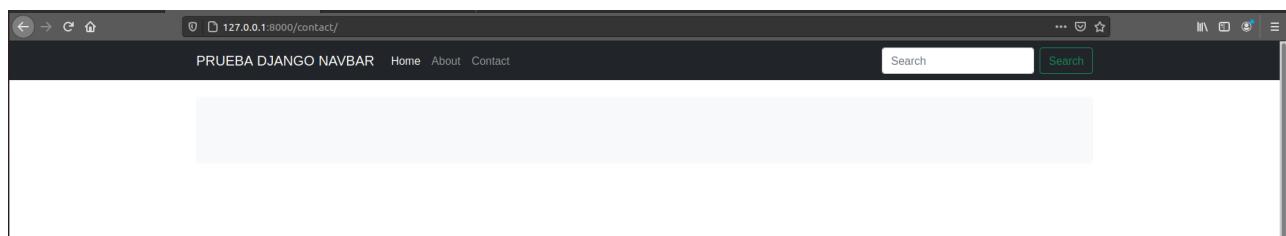


```

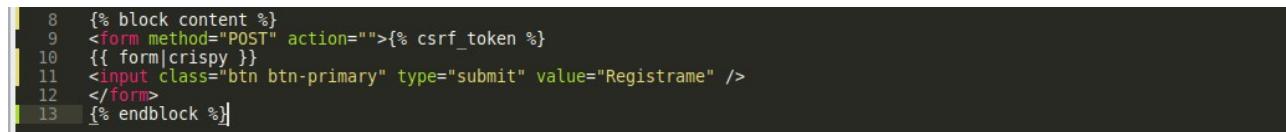
1  {% extends "base.html" %} 
2  {% load crispy_forms_tags %} 
3 
4  {{ titulo }}<br/> 
5  {{ request.user }}<br/> 
6  <hr/> 
7  <br/> 
8 
9  <form method="POST" action="">{{ csrf_token }} 
10 {{ form|crispy }} 
11 <input type="submit" value="Registrate" /> 
12 </form>

```

Y nuestra página de contacto se vería así:



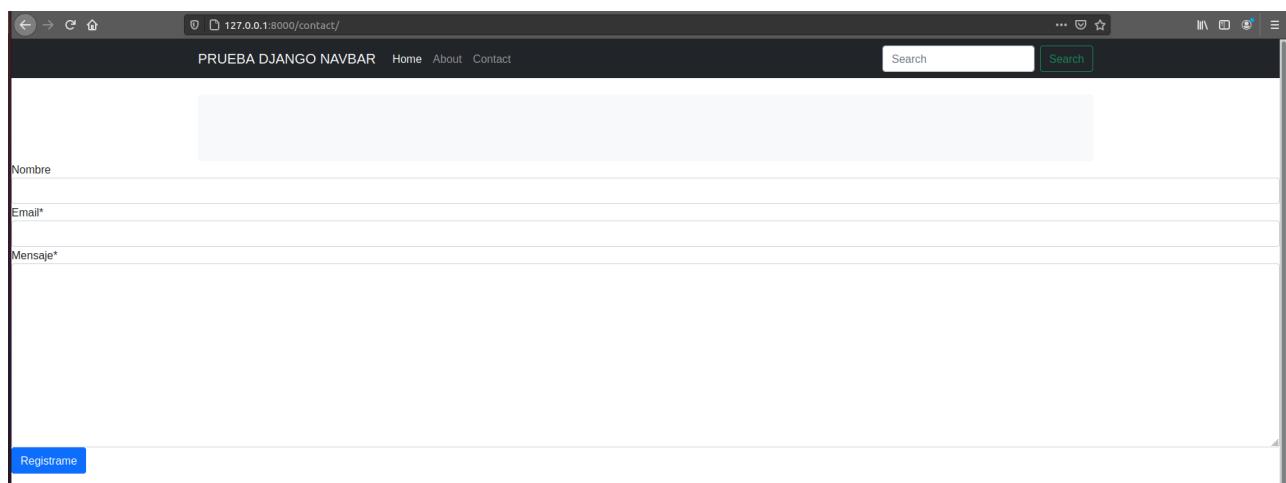
Esta página no tiene ningún tipo de formulario. Ahora lo iremos implementando.



```

8  {% block content %} 
9  <form method="POST" action="">{{ csrf_token }} 
10 {{ form|crispy }} 
11 <input class="btn btn-primary" type="submit" value="Registrate" /> 
12 </form> 
13  {% endblock %}

```



Vamos a usar el sistema de cuadrículas de *bootstrap*:

```
8  {% block content %} 
9  <div class="row">
10 <div class="col-6">
11 <form method="POST" action="">{% csrf_token %}
12 {{ form|crispy }}
13 <input class="btn btn-primary" type="submit" value="Registrate" />
14 </form>
15 </div>
16 {% endblock %}
```

The screenshot shows a web browser window with the URL 127.0.0.1:8000/contact/. The page title is PRUEBA DJANGO NAVBAR. The content area contains a form with three input fields: 'Nombre', 'Email*', and 'Mensaje*'. Below the inputs is a blue 'Registrate' button. The form is styled using Bootstrap's grid system, specifically a row with a single col-6 column containing the form.

Vamos a centrar el formulario un poco. Nosotros al usar *bootstrap 4* y karlita la versión 3, el código tiene otra estructura:

```
8  {% block content %} 
9  <div class="row justify-content-evenly">
10 <div class="col-6">
11 <form method="POST" action="">{% csrf_token %}
12 {{ form|crispy }}
13 <input class="btn btn-primary" type="submit" value="Registrate" />
14 </form>
15 </div>
16 {% endblock %}
```

Y finalmente se ve así:

The screenshot shows the same web browser window with the URL 127.0.0.1:8000/contact/. The form is now centered horizontally within the row and col-6 column, as indicated by the 'justify-content-evenly' class in the CSS. The layout remains otherwise identical to the previous screenshot.

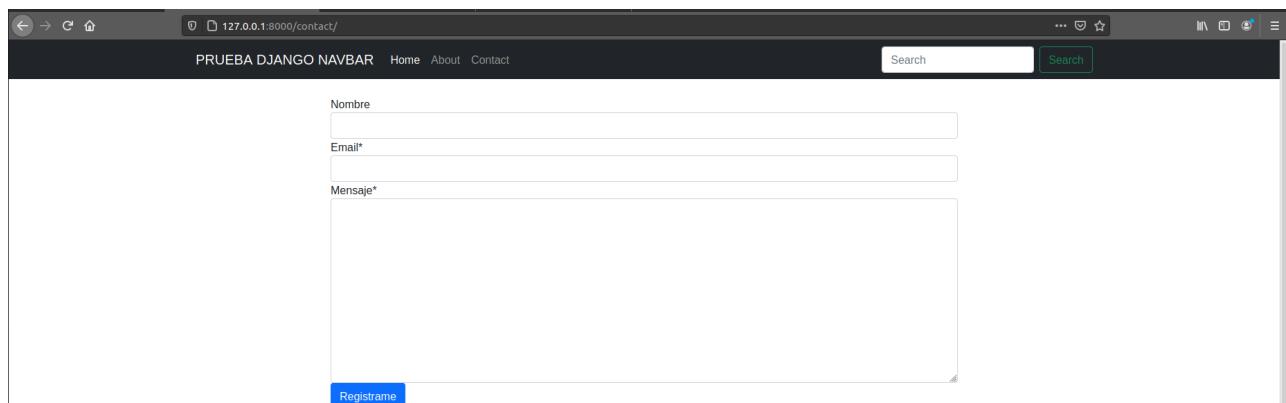
Todo esto conlleva que este bloque de aquí en ‘base.html’ sobre:

```
<main class="container">
  {% block container-fluid %}
```

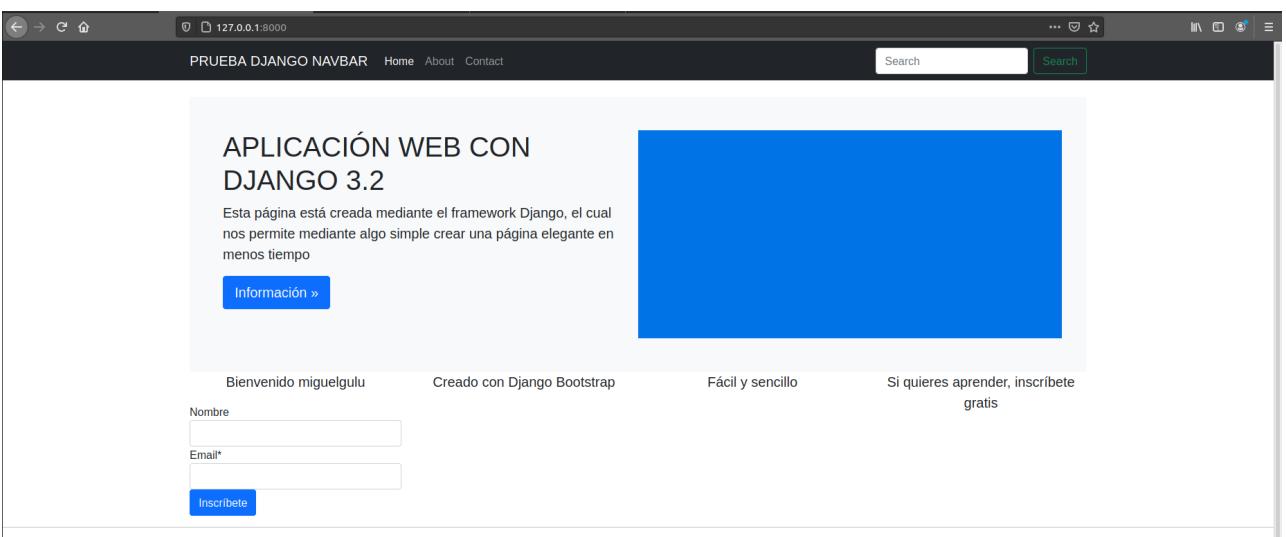
Y llevamos el apartado de arriba sobrante a inicio y quitamos el *content* de *container*:

```
7  {% block container %}
8    <div class="bg-light p-5 rounded">
9      <div class='row'>
10        <div class='col-sm-6'>
11          <h1>APLICACION WEB CON DJANGO 3.2</h1>
12          <p class="lead">Esta página está creada mediante el framework Django, el cual nos permite mediante algo simple crear una página elegante en menos tiempo</p>
13          <a class="btn btn-lg btn-primary" href="/docs/5.0/components/navbar/" role="button">Información &raquo;</a>
14        </div>
15        <div class='col-sm-6' style='background-color:#0073e6; height:300px;'></div>
16      </div>
17    </div>
18  {% endblock %}
```

Y nuestro *container fluid* desaparece:



Pero nuestro *container content* sigue en inicio:



Ahora en nuestro formulario vamos a añadir un título para que quede más presentable...
Nos vamos a nuestro ‘forms.py’ y añadimos lo siguiente:

```

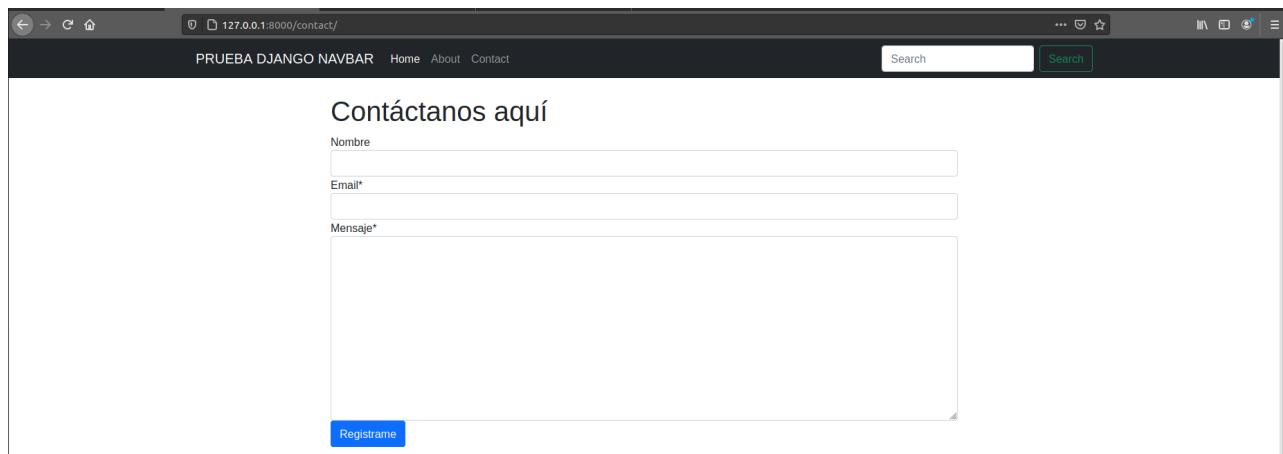
8   {% block content %}
9     <div class="row justify-content-evenly">
10    <div class="col-6">
11      {% if titulo %}
12        <h1>{{ titulo }}</h1>
13      {% endif %}
14      <form method="POST" action="">{{ csrf_token }}
15      {{ form|crispy }}
16      <input class="btn btn-primary" type="submit" value="Regístrate" />
17    </form>
18  </div>
19  {% endblock %}
```

Y vamos a crear la variable de título en nuestra función en ‘views.py’:

```

42 def contact(request):
43     titulo = "Contáctanos aquí"
44     form = ContactForm(request.POST or None)
45     if form.is_valid():
46         #for key, value in form.cleaned_data.items():
47         #    print (key, value)
48         #for key in form.cleaned_data:
49         #    print (key)
50         #    print(form.cleaned_data.get(key))
51         form_nombre = form.cleaned_data.get("nombre")
52         form_email = form.cleaned_data.get("email")
53         form_mensaje = form.cleaned_data.get("mensaje")
54         asunto = 'Formulario de Contacto'
55         email_from = settings.EMAIL_HOST_USER
56         email_to = [email_from, "test@gmail.com"]
57         mensaje_email = "%s: %s enviado por %s" %(form_nombre, form_mensaje, form_email)
58         #print(email, mensaje, nombre)
59         send_mail(asunto,
60                   mensaje_email,
61                   email_from,
62                   [email_to],
63                   fail_silently=False
64                 )
65     context = {
66         "form": form,
67         "titulo": titulo,
68     }
69     return render(request, "forms.html", context)
```

Y finalmente, tenemos nuestro título en el formulario de contacto!

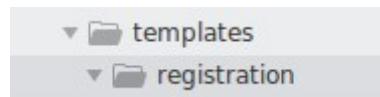


Si lo queremos alinear vusta con añadir el ‘align center’...

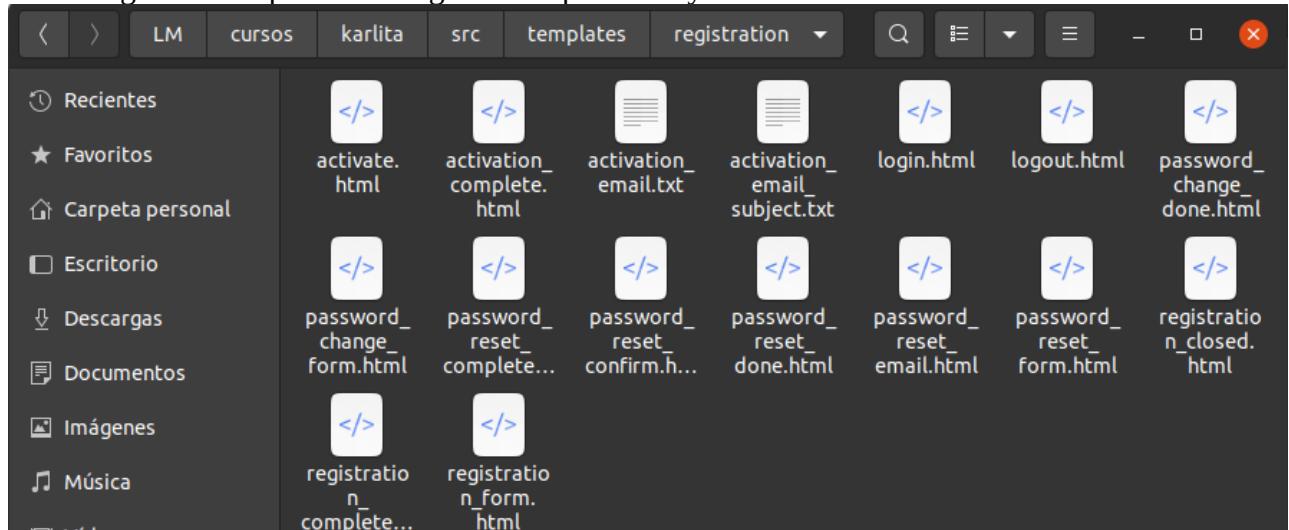
The screenshot shows a web browser window with the URL `127.0.0.1:8000/contact/`. The page title is "PRUEBA DJANGO NAVBAR". The main content is a form titled "Contáctanos aquí". It contains three input fields: "Nombre", "Email*", and "Mensaje*". Below the form is a blue button labeled "Regístrate". The browser's address bar, header, and search bar are also visible.

30. DJANGO REGISTRATION REDUX 1

Ahora vamos a utilizar el recurso de Registration Redux el cual nos va a permitir añadir registros básicos, login y autenticación de usuario. Todo esto nos puede ayudar a reducir código. En primer lugar, y muy importante, necesitamos añadir unas plantillas antes de instalarlo. Para ello, creamos la carpeta ‘registration’ dentro de templates:



Y descargamos de su repositorio de github las plantillas y añadirlas...



Y ahora procederemos con la instalación:

```
^C(venv) alunsiito@alunsiito:~/Documentos/LM/cursos/karlita/src$ pip3 install django-registration-redux
Collecting django-registration-redux
  Downloading django_registration_redux-2.9-py2.py3-none-any.whl (209 kB)
|██████████| 209 kB 1.5 MB/s
Installing collected packages: django-registration-redux
Successfully installed django-registration-redux-2.9
(venv) alunsiito@alunsiito:~/Documentos/LM/cursos/karlita/src$
```

Y una vez instalado, tenemos que añadirlo a nuestros settings en INSTALLED_APP, además de una app de django ‘sites’:

```

38 INSTALLED_APPS = [
39     #Django
40     'django.contrib.sites',
41     'django.contrib.admin',
42     'django.contrib.auth',
43     'django.contrib.contenttypes',
44     'django.contrib.sessions',
45     'django.contrib.messages',
46     'django.contrib.staticfiles',
47     #Terceros
48     'boletin.apps.BoletinConfig',
49     'crispy_forms',
50     'registration',
51 ]

```

Además de añadir las siguientes variables...

```

65 ACCOUNT_ACTIVATION_DAYS = 7 # One-week activation window; you may, of course, change this.
66 REGISTRATION_AUTO_LOGIN = True # Automatically log the user in.
67

```

Esto significa se refiere al número de días que tiene un usuario para registrarse y para automáticamente cuando pinche el el botón de sesión este haga log automáticamente.

Finalmente, debemos hacer el respectivo *migrate*:

```

(venv) alunsiito@alunsiito:~/Documentos/LM/cursos/karlita/src$ python3 manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, boletin, contenttypes, registration, sessions, sites
Running migrations:
  Applying registration.0001_initial... OK
  Applying registration.0002_registrationprofile_activated... OK
  Applying registration.0003_migrate_activatestatus... OK
  Applying registration.0004_supervisedregistrationprofile... OK
  Applying registration.0005_activation_key_sha256... OK
  Applying sites.0001_initial... OK
  Applying sites.0002_alter_domain_unique... OK
(venv) alunsiito@alunsiito:~/Documentos/LM/cursos/karlita/src$ 

```

Se han añadido las apps con éxito.

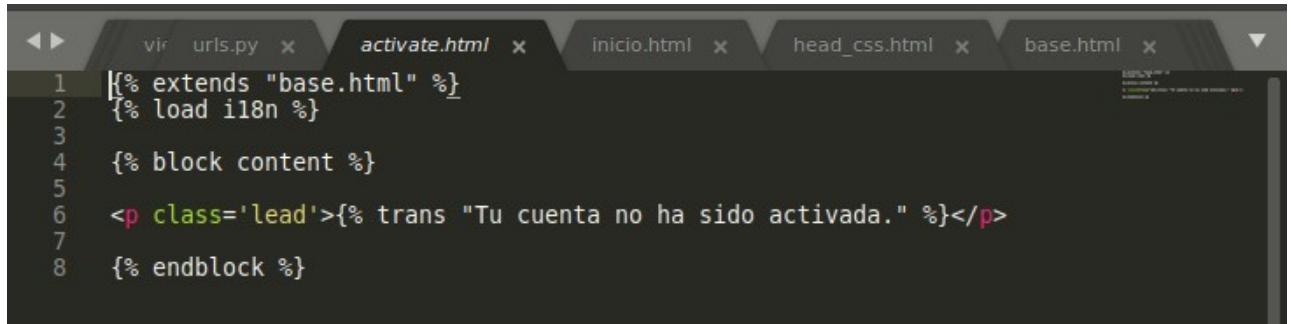
Falta añadir la respectiva url (hace falta hacer la importación de *include* de *django.conf.urls*):

```

25 urlpatterns = [
26     path('admin/', admin.site.urls),
27     path('', views.inicio, name='inicio'),
28     path('contact/', views.contact, name='contact'),
29     path('about/', about, name='about'),
30     path('accounts/', include('registration.backends.default.urls'))]
31 ]

```

Vamos a asegurarnos de que nuestros archivos nuevos sean la extensión de base.html:

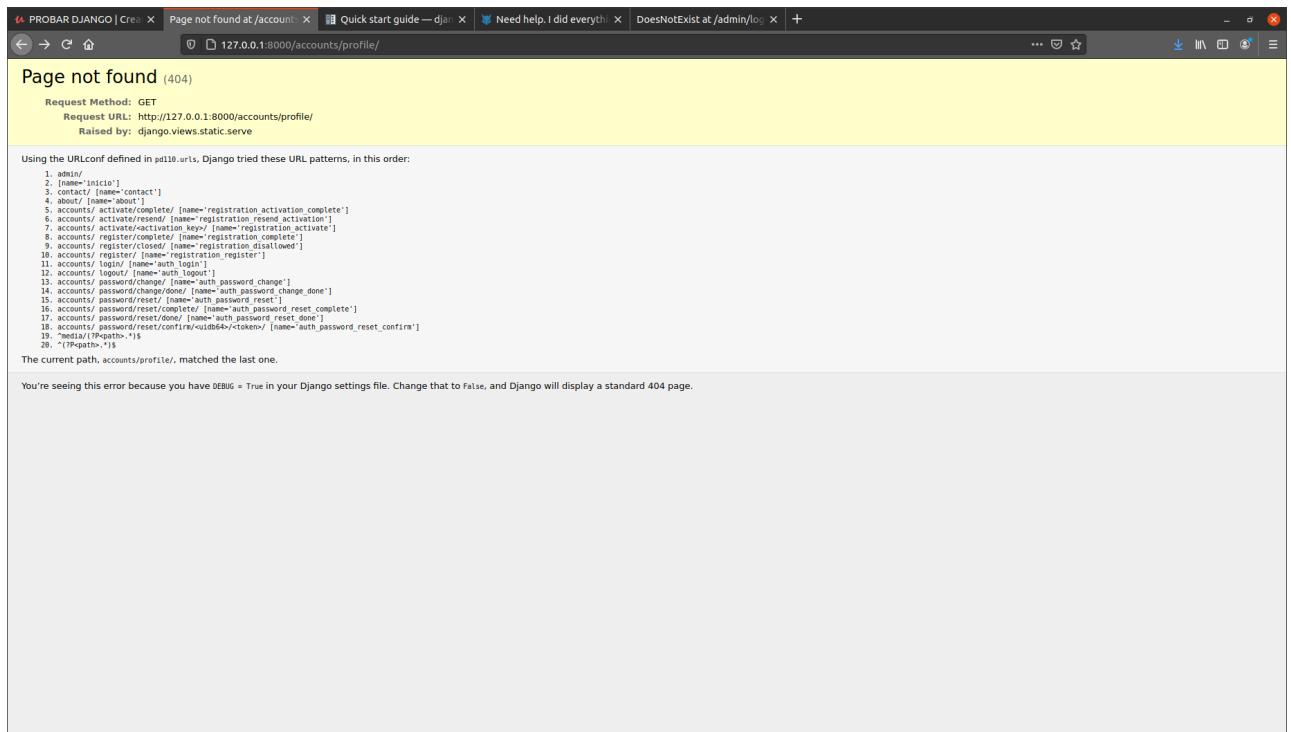


```

1  {% extends "base.html" %}
2  {% load i18n %}
3
4  {% block content %}
5
6  <p class='lead'>{% trans "Tu cuenta no ha sido activada." %}</p>
7
8  {% endblock %}

```

Y si hacemos runserver y abrimos en el navegador la url con `accounts/register/ ...`



Nos da como página no encontrada (404) y nos redirige a `profiles/`. Esto se debe a que al estar logueado en admin se nos autoreconoce y no hace falta el `register`.

Si cerramos sesión y volvemos a la página...

The screenshot shows a browser window with multiple tabs open at the top. The active tab is '127.0.0.1:8000/accounts/register/'. The page title is 'PRUEBA DJANGO NAVBAR'. The main content is a registration form with the heading 'Registrarte Gratis!'. It includes fields for 'Nombre de usuario*', 'Correo Electrónico*', 'Contraseña*', and 'Contraseña (confirmación)*'. Below the password fields is a note: 'Para verificar, introduzca la misma contraseña anterior.' A blue button labeled 'Registrarme' is at the bottom left, and a link 'Ya tienes cuenta? Iniciar Sesión.' is at the bottom right.

Vamos a darle un poco más de estilo al formulario:

This screenshot shows the same registration form as the previous one, but with a different CSS style applied. The heading '¡Registrar Ahora!' is in a larger, bold font. The input fields have rounded corners and a slight shadow. The error message below the email field is smaller and less prominent. The overall layout is cleaner and more modern.

31. DJANGO REGISTRATION REDUX 1

Nos tocará en este vídeo diferenciar las *urls* que corresponde a cada archivo. Para ello, si en la *url* de *accounts* escribimos cualquier cosa, nos saldrá todas las posibles *urls* que podríamos haber escrito:

The screenshot shows a browser window with the URL `http://127.0.0.1:8000/accounts/ksadjfkladsnf`. The title bar says "Page not found (404)". The page content lists all the URL patterns defined in `urls.py`, including various registration and login paths like `/accounts/activate/`, `/accounts/activate/complete/`, `/accounts/activate/review/`, `/accounts/activate/complete/key/`, `/accounts/register/`, `/accounts/register/complete/`, `/accounts/register/disallowed/`, `/accounts/login/`, `/accounts/logout/`, `/accounts/password/change/`, `/accounts/password/change/done/`, `/accounts/password/reset/`, `/accounts/password/reset/complete/`, `/accounts/password/reset/done/`, `/accounts/password/reset/confirm/`, and `/accounts/password/reset/confirm/uidb64/:token/`. A note at the bottom says "You're seeing this error because you have DEBUG = True in your Django settings file. Change that to False, and Django will display a standard 404 page."

Nos dirigimos a `/accounts/login/`:

The screenshot shows a browser window with the URL `http://127.0.0.1:8000/accounts/login/`. The title bar says "DoesNotExist at /accounts/login/". The page content shows the stack trace for the error, indicating that a site matching the query does not exist. The stack trace includes details about the Python environment, including the path `/home/alunusito/Documentos/LM/cursos/karlita/venv/lib/python3.8/site-packages/django/db/models/query.py` and line 435. The server time is listed as "Thu, 22 Apr 2021 09:29:32 +0000".

Esto se soluciona editando el `setting.py`:

```
65 ACCOUNT_ACTIVATION_DAYS = 7 # One-week activation window; you may, of
66 REGISTRATION_AUTO_LOGIN = True # Automatically log the user in.
67 SITE_ID = 1
68
```

Y si recargamos la página, nos saldrá correctamente:

PRUEBA DJANGO NAVBAR Home About Contact Search Search

Iniciar Sesión

Nombre de usuario*
miguelgulu

Contraseña*

Iniciar sesión

Has olvidado tu contraseña? [Restablecer!](#)

No tienes cuenta? [Registrarte!](#)

Con esa línea que hemos añadido lo que hemos hecho es activar la aplicación *site*, la cual estaba incluida pero no activada.

Si entramos en nuestro *admin*, tendremos nuestros sitios:

Administración de Django

Bienvenido, MIGUELGULU VER EL SITIO / CAMBIAR CONTRASEÑA / CERRAR SESIÓN

Sitio administrativo

AUTENTICACIÓN Y AUTORIZACIÓN

- Grupos [Añadir](#) [Modificar](#)
- Usuarios [Añadir](#) [Modificar](#)

BOLETÍN

- Registrados [Añadir](#) [Modificar](#)

REGISTRATION

- Perfiles de registro [Añadir](#) [Modificar](#)

SITIOS

- Sitios [Añadir](#) [Modificar](#)

Acciones recientes

Mis acciones

- x usuario@usuario.edu Registrado
- + validacion@email.edu Registrado
- + validacion@eduardo.com Registrado

Y si pinchamos en sitios tendremos un dominio por defecto de nuestra página web.

```

navbar.html x forms.py x forms.html x settings.py x activation_email.txt x
1  {% load i18n %}_
2
3  {% trans "Activa tu cuenta en" %} {{ site.name }}:
4
5  Hola,
6  Haz click en el enlace para activar tu cuenta.
7  http://{{ site.domain }}{% url 'registration_activate' activation_key %}
8
9  {% blocktrans %}Enlace válido durante {{ expiration_days }} días.{%
10 endblocktrans %}
11 -Team CFE

```

En nuestro archivo de 'activation_email.txt' está la línea que indica nuestra URL definida por el dominio del sitio.

A continuación, vamos a registrar un nuevo usuario en ‘register’:

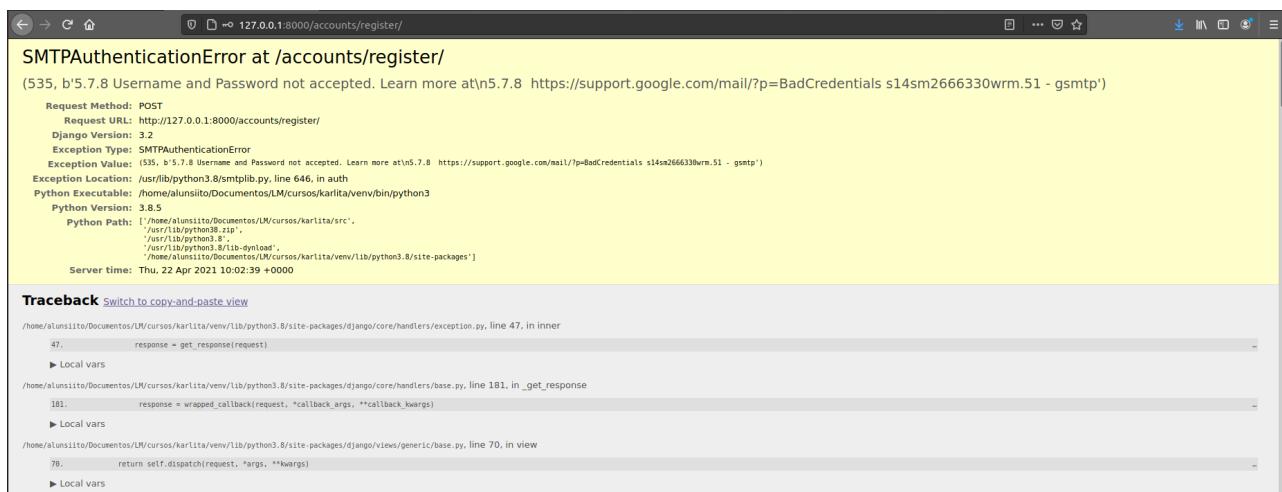
The screenshot shows a registration form with the following fields:

- Nombre de usuario***: pruebesita
- Correo Electrónico***: pruebesita@gmail.com
- Contraseña***: (password masked)
- Contraseña (confirmación)***: (password masked)

Below the password fields, there is a note: "Para verificar, introduzca la misma contraseña anterior."

A blue "Registrarme" button is at the bottom.

Y si le damos a ‘Regístrate’...



Nos sale el mismo error que nos salió en su momento como usamos un correo que no existe en la base de datos de google. En cambio, si nos vamos al admin site y a perfiles de registro tenemos el usuario a a vista:

The screenshot shows the 'Administración de Django' interface with the 'Perfiles de registro' section selected. The page title is 'Selección de perfil de registro a modificar'. It lists two users:

Acción:	Usuario	Activación Key Expired
Ir	pruebesita	False

At the bottom, it says '1 perfil de registro'.

Modificar perfil de registro
Registration information for pruebesita

Usuario: 3

Clave de activación: 22e13ff4bda18990d06ef98d5befd2a8e4db7

Activated

Eliminar **Guardar y añadir otro** **Guardar y continuar editando** **GUARDAR**

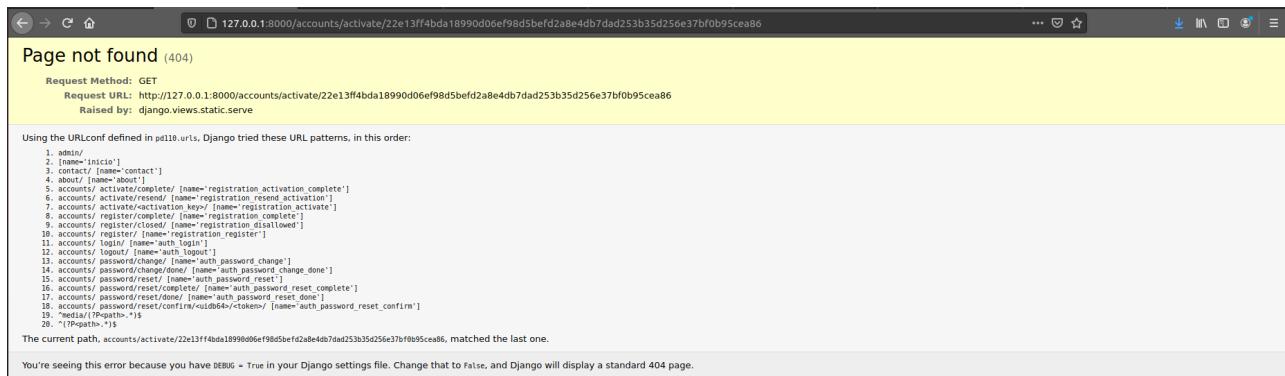
Como vemos, no está activado pero si tenemos su clave de activación.

Vamos a buscar a continuación la URL de activación. Para ello, nos vamos al repositorio de [github](#) que se indica y analizamos el archivo ‘urls.py’:

```
40     path('activate/<activation_key>',
41         ActivationView.as_view(),
42         name='registration_activate'),
43     path('register/complete/',
44         TemplateView.as_view(template_name='registration/registration_complete.html'),
```

Aquí esta.

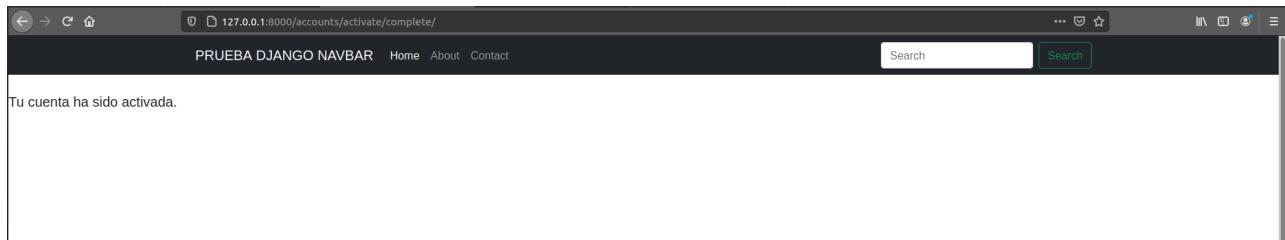
Esta url está compuesta por /accounts/activate/[key de activacion]. Entonces para acceder a esa url cogemos la clave de activación que nos salía en el perfil registrado. Vamos a hacer la prueba:



```
haz click en el enlace para activar tu cuenta.  
http://{{ site.domain }}% url 'registration_activate' activation_key %}
```

Ambas representan la misma url.

Y si la clave es correcta, saldrá la vista de que la cuenta ha sido activada.



Vamos a comprobar que ha sido activada en nuestro admin:

A screenshot of the Django Admin interface. The left sidebar shows 'Perfiles de registro' selected. The main area is titled 'Seleccióne perfil de registro a modificar'. It lists one item: 'pruebesita' with status 'ACTIVATION KEY EXPIRED' and 'True'. There is a search bar at the top and a 'AÑADIR PERFIL DE REGISTRO' button.

Y vemos como nos sale que la clave de activación ha expirado ya que se ha usado y ha activado la cuenta:

A screenshot of the Django Admin interface showing the 'Modificar perfil de registro' (Edit registration profile) form. It displays 'Registration information for pruebesita'. Fields include 'Usuario' (set to 3) and 'Clave de activación' (set to 22e13ff4bda18990d06ef98d5befd2a8e4db7). A checkbox 'Activated' is checked. At the bottom are buttons for 'Eliminar', 'Guardar y añadir otro', 'Guardar y continuar editando', and 'GUARDAR'.

Seguimos con la comprobación de los *login* y *logout*. Para ello empezamos yendo a la url que corresponde a la vista de *login*:

Iniciar Sesión

Nombre de usuario*

pruebesita

Contraseña*

password

[Iniciar sesión](#)

Has olvidado tu contraseña? [Restablecer!](#)

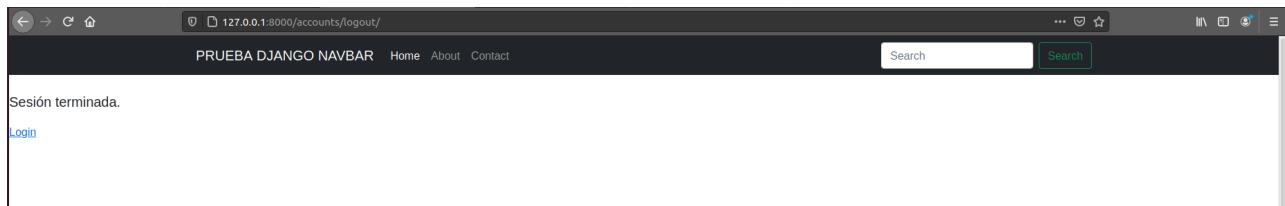
No tienes cuenta? [Registrarte!](#)

Probaremos nuestro nuevo usuario creado...



Y nos manda a la url de *profile* y nos dice que no ha sido encontrada ya que no lo tenemos configurado para que funcione.

Vamos a probar también la vista de *logout*:



A continuación, iniciaremos sesión con nuestro superusuario:



Nos pasa exactamente lo mismo pero, si vamos a nuestro admin...

The screenshot shows the Django Admin interface at the URL `127.0.0.1:8000/admin/`. The left sidebar has sections for 'AUTENTICACIÓN Y AUTORIZACIÓN' (Groups, Users), 'BOLETIN' (Registrados), 'REGISTRATION' (Perfiles de registro), and 'SITIOS' (Sitios). The main content area shows a table of registered users:

Usuario	Correo electrónico*	Estado
usuario@usuario.edu	usuario@usuario.edu	Registrado
validacion@email.edu	validacion@email.edu	Registrado
validacion@eduardo.com	validacion@eduardo.com	Registrado
email@email.com	email@email.com	Registrado
test@gmail.com	test@gmail.com	Registrado

A sidebar on the right lists 'Acciones recientes' (Recent actions) with entries like 'usuario@usuario.edu Registrado', 'validacion@email.edu Registrado', etc.

Tendremos nuestra sesión ya iniciada.

También, gracias a *registration redux*, tenemos la opción de hacer reset a nuestra contraseña. Para ello, nos dirigimos a su correspondiente url.

The screenshot shows a password reset form at the URL `127.0.0.1:8000/accounts/password/reset/`. The page title is 'Restablecer Contraseña'. It has a field labeled 'Correo electrónico*' and a blue 'Enviar' button.

Gracias a nuestro archivo de '*password_reset_email.html*', a través de nuestro dominio se enviará un correo para poder restablecer la contraseña.

Si completáramos todo, nos saldría la vista para informarnos de que la contraseña ha sido restablecida:

The screenshot shows a confirmation message at the URL `127.0.0.1:8000/accounts/password/reset/complete/`. The message reads 'Contraseña ha sido restablecida correctamente.' and includes a blue 'Iniciar sesión' link.

Y si hacemos click en 'Iniciar sesión', nos mandaría a la dirección para hacer login de nuevo.

Para terminar, vamos a probar a registrar un usuario que existe y comprobar que no nos lo da por válido porque ya existe:

The screenshot shows a web browser window with the URL `127.0.0.1:8000/accounts/register/`. The page title is "PRUEBA DJANGO NAVBAR". The main heading is "¡Regístrate Ahora!". The "Nombre de usuario*" field contains "pruebesita", which is highlighted with a red border and has a small red circular icon with a question mark next to it. Below the field, a red error message reads "Ya existe un usuario con ese nombre de usuario.". There are three input fields below: "Correo Electrónico*" with value "pruebesita@gmail.com", "Contraseña*" with a placeholder, and "Contraseña (confirmación)". A note below the password fields says "Para verificar, introduzca la misma contraseña anterior." A blue "Registrarme" button is at the bottom. At the very bottom of the page, there is a link "Ya tienes cuenta? [Iniciar Sesión.](#)"

El mensaje de error funciona correctamente.

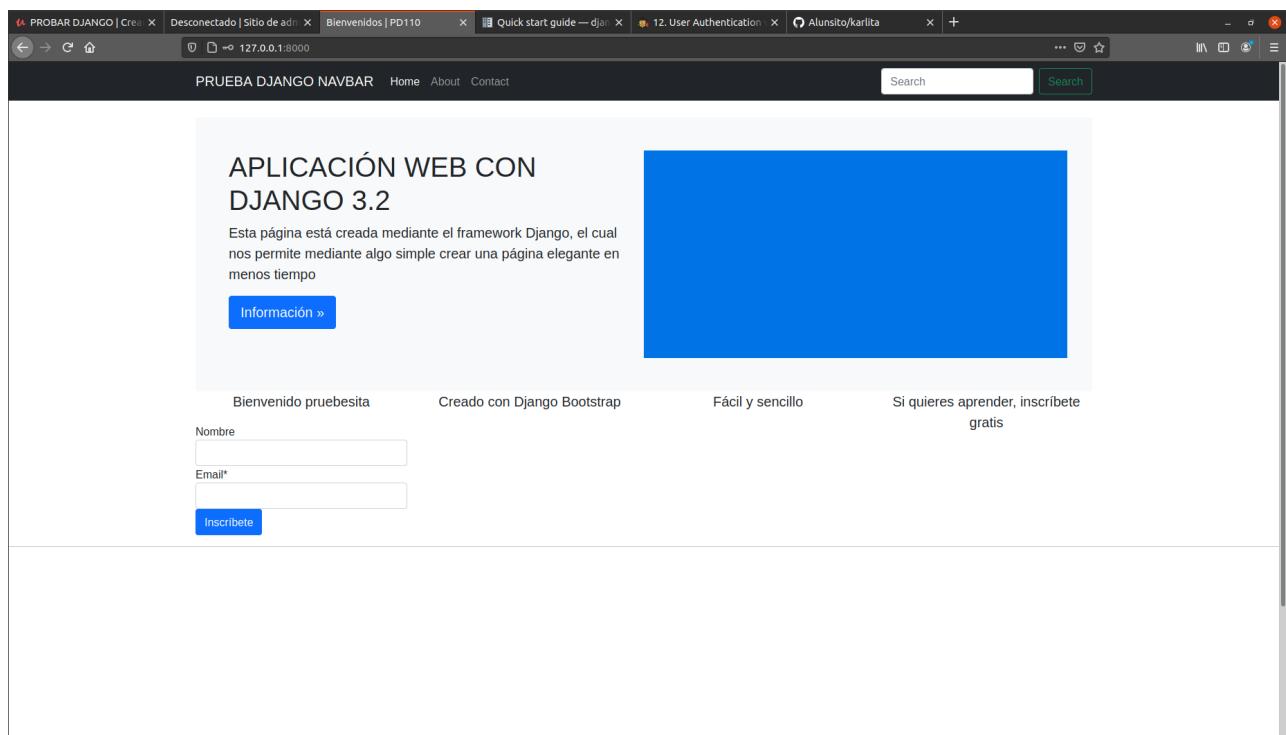
32. CAMBIAR URL REDIRECT DESPUÉS DE UN LOGIN

En este breve video vamos a arreglar el problema que pasa cuando iniciamos sesión, el cual devuelve el error 404 porque no está la página del perfil definida. Este redirect se hace yendo a nuestro ‘setting.py’ tendremos que añadir la siguiente variable:

```
67 SITE_ID = 1
68 LOGIN_REDIRECT_URL = '/'
69
```

Y ahí elegimos la url destino de cuando iniciamos sesión...

Y si iniciamos sesión con el usuario que creamos en el vídeo anterior, nos redirigirá a la página de inicio:



Si nos fijamos en el mensaje de Bienvenido, podemos ver como sale el usuario que creamos.

33. AUTENTICACIÓN PARA ENLACES EN EL NAVBAR

En esta parte vamos a dedicarnos a completar las direcciones que corresponderían a los enlaces de nuestra barra de navegación. En el caso de ‘Home’ ya funciona, pero otros no.

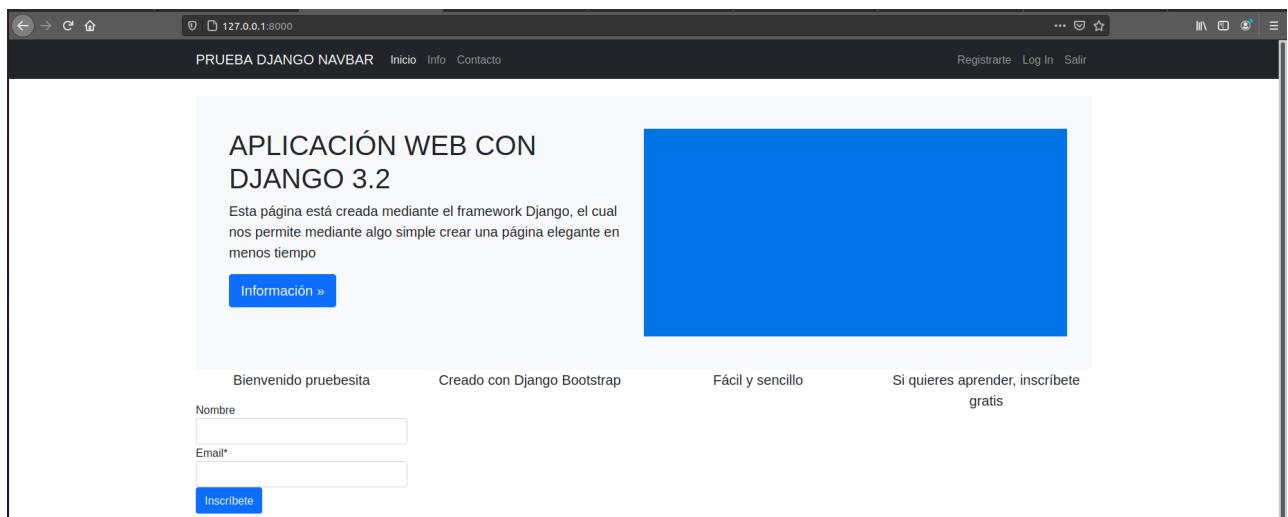
Para empezar, nos vamos a nuestra página de ‘navbar.html’. Vamos a adaptar el navbar un poco más a como se usa en el curso para que podamos hacer la demostración:

```

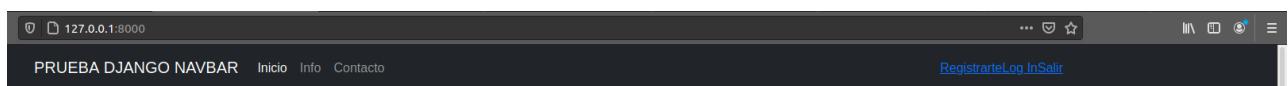
1 <nav class="navbar navbar-expand-md navbar-dark bg-dark mb-4">
2   <div class="container">
3     <a class="navbar-brand" href="/">PRUEBA DJANGO NAVBAR</a>
4     <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarCollapse" aria-controls="navbarCollapse" aria-expanded="false" aria-label="Toggle navigation">
5       <span class="navbar-toggler-icon"></span>
6     </button>
7     <div class="collapse navbar-collapse" id="navbarCollapse">
8       <ul class="navbar-nav me-auto mb-2 mb-md-0">
9         <li class="nav-item">
10           <a class="nav-link active" aria-current="page" href="/">Inicio</a>
11         </li>
12         <li class="nav-item">
13           <a class="nav-link" href="{% url 'about' %}">Info</a>
14         </li>
15         <li class="nav-item">
16           <a class="nav-link" href="{% url 'contact' %}" tabindex="-1" aria-disabled="true">Contacto</a>
17         </li>
18       </ul>
19       <ul class="nav navbar-nav navbar-right">
20         <li class="nav-item"><a class="nav-link" href="#">Registrarte</a></li>
21         <li class="nav-item"><a class="nav-link" href="#">Log In</a></li>
22         <li class="nav-item"><a class="nav-link" href="#">Salir</a></li>
23       </ul>
24     </div>
25   </div>
26 </nav>

```

Lo cual hace que se vea así:



A diferencia de karlita, he tenido que añadir las clases ‘nav-item’ y ’nav-link’, si no, se vería así:

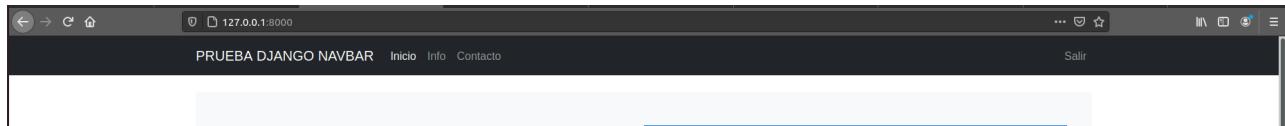


Si hacemos click en cualquiera de los elementos nos va a enviar a la misma página de inicio. Además, lo que nos interesa hacer es que si hemos iniciado sesión nos de la opción de cerrarla y viceversa.

Para ello habrá que añadir un condicional:

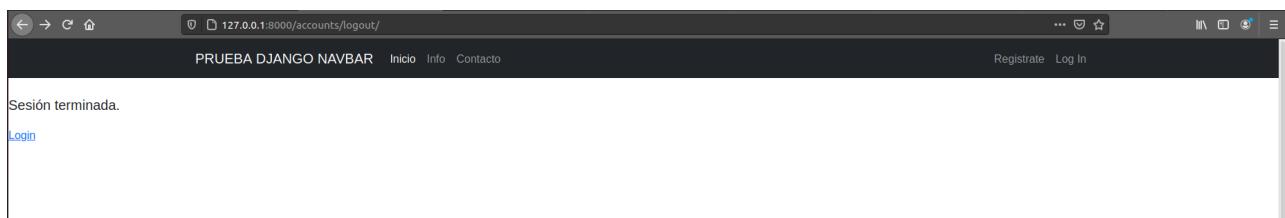
```
19      <ul class="nav navbar-nav navbar-right">
20      {% if request.user.is_authenticated %}
21      <li class="nav-item"><a class="nav-link" href="">Salir</a></li>
22      {% else %}
23      <li class="nav-item"><a class="nav-link" href="">Registrate</a></li>
24      <li class="nav-item"><a class="nav-link" href="">Log In</a></li>
25      {% endif %}
26      </ul>
27      </div>
28  </div>
29  </nav>
```

Guardamos y lo comprobamos...



Como hemos iniciado sesión antes con ‘pruebesita’ solo tenemos la opción de cerrar sesión.

Si cierro sesión en *accounts/logout*:



Ya nos sale arriba a la derecha la opción de registrarme e iniciar sesión.

Ahora toca hacer que funcione esos enlaces:

```
21      <li class="nav-item"><a class="nav-link" href="{% url 'auth_logout' %}">Salir
22      </a></li>
23      {% else %}
24      <li class="nav-item"><a class="nav-link" href="{% url 'registration_register' %}">Registrate</a></li>
25      <li class="nav-item"><a class="nav-link" href="{% url 'auth_login' %}">Log In
26      </a></li>
27      {% endif %}
28      </ul>
```

Y comprobamos que funcionan los enlaces:

The screenshot shows a registration form titled "¡Regístrate Ahora!" (Register Now!). The form includes fields for "Nombre de usuario*" (Username*), "Correo Electrónico*" (Email*), "Contraseña*" (Password*), and "Contraseña (confirmación)*" (Password confirmation*). A note below the password fields says "Para verificar, introduzca la misma contraseña anterior." (To verify, enter the same password again.). A "Rellene este campo." (Fill this field.) message is displayed above the password confirmation field. A "Regístrate" (Register) button is at the bottom. The URL in the address bar is 127.0.0.1:8000/accounts/register/.

The screenshot shows a login form titled "Iniciar Sesión" (Log In). It has fields for "Nombre de usuario*" (Username*) containing "miguelgulu" and "Contraseña*" (Password*) containing "*****". A "Iniciar sesión" (Log In) button is highlighted in blue. Below the form are links for password recovery ("Has olvidado tu contraseña? Restablecer!") and new user registration ("No tienes cuenta? Registrarse!"). The URL in the address bar is 127.0.0.1:8000/accounts/login/.

The screenshot shows a confirmation page after logging out. It displays the message "Sesión terminada." (Session ended.) and a "Login" link. The URL in the address bar is 127.0.0.1:8000/accounts/logout/.

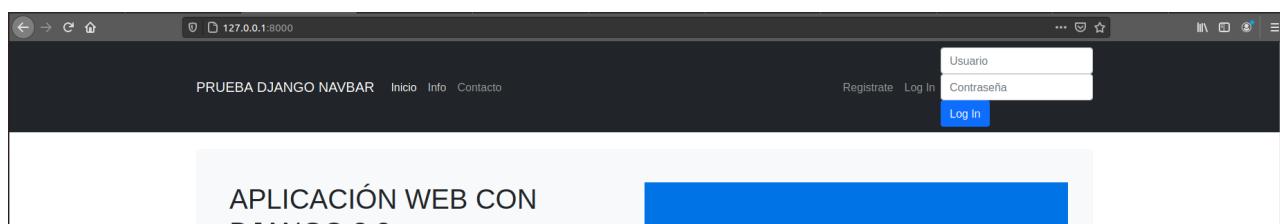
34. FORMULARIO DE LOGIN EN LA NAVBAR

Para tener un poco más accesible la opción de hacer *login* sin tener que entrar en otra página vamos a enfocarnos en añadirla a nuestra barra de navegación.

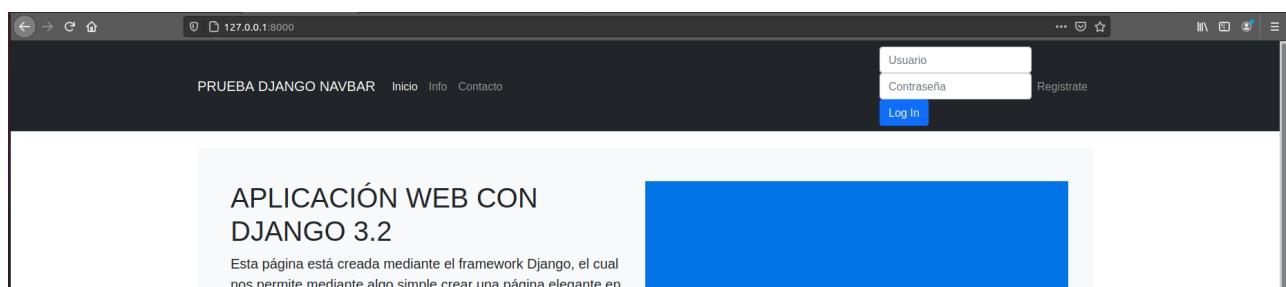
Nos dirigimos al archivo de navbar y vamos a añadir un nuevo form ahí:

```
28      <form class="navbar-form navbar-right" method="POST" action="">{% csrf_token %}</form>
29      <div class="form-group"></div>
30          <input type="text" class="form-control" name="username" placeholder="Usuario"/>
31          <div class="form-group"></div>
32              <input type="text" class="form-control" name="password" placeholder="Contraseña" />
33          <button type="submit" class="btn btn-primary">Log In</button>
34      </div>
35  </div>
36 </nav>
```

Y así se verá nuestro formulario:

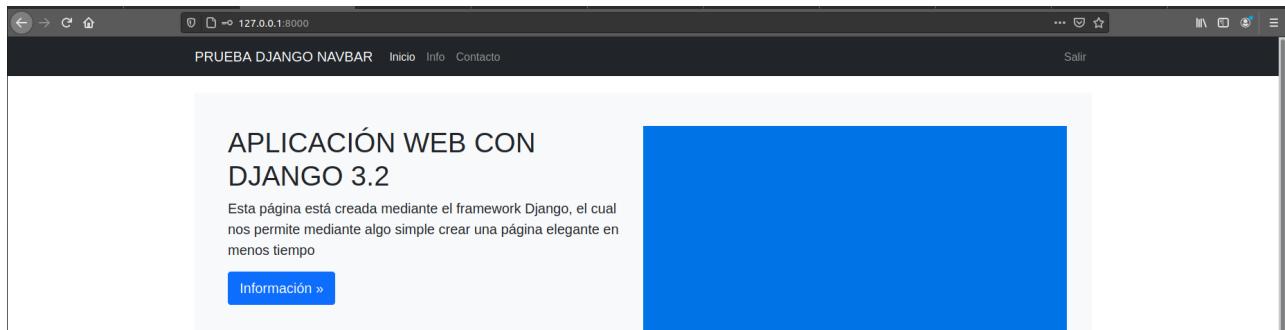


Vamos a ordenarlo un poco y quitar el duplicado de login:



Y ahora debemos añadir la condición para que solo aparezca si no se está autenticado:

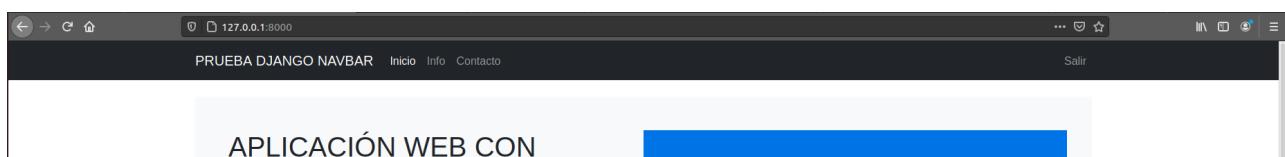
```
{% if not request.user.is_authenticated %}
<form class="navbar-form navbar-right" method="POST" action="">{% csrf_token %}</form>
<div class="form-group"></div>
    <input type="text" class="form-control" name="username" placeholder="Usuario" />
<div class="form-group"></div>
    <input type="text" class="form-control" name="password" placeholder="Contraseña" />
<button type="submit" class="btn btn-primary">Log In</button>
</form>
{% endif %}
```



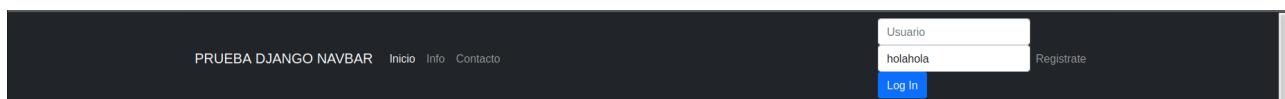
Para loguearnos teníamos que ir a la dirección de login hasta ahora, pero añadiendo en ‘action’ la url de autenticación, sería más que suficiente:

```
<form class="navbar-form navbar-right" method="POST" action="{% url 'auth_login' %}">
  {% csrf_token %}
    <div class="form-group"></div>
```

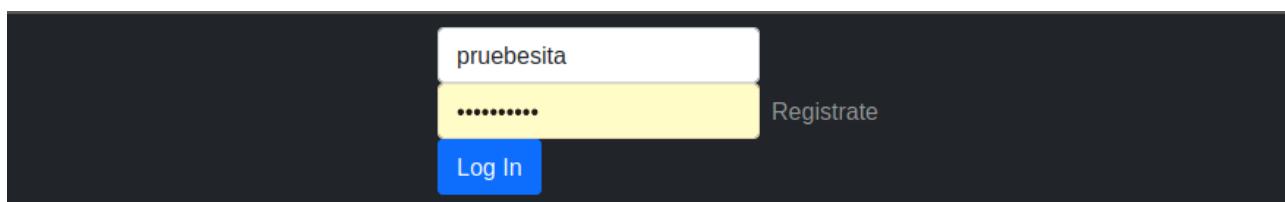
Lo probamos...



Un problema de todo esto es que si rellenamos ese formulario la contraseña es visible:



Para cambiar eso, en el tipo de texto debemos poner ‘password’:



Si escribo una contraseña incorrecta nos mandará al login...

The screenshot shows a browser window with the URL `127.0.0.1:8000/accounts/login/`. At the top, there is a navigation bar with the text "PRUEBA DJANGO NAVBAR" and links for "Inicio", "Info", and "Contacto". On the right side of the header, there is a "Registrate" button. Below the header, a "Log In" form is displayed. The "Nombre de usuario*" field contains "pruebesita". The "Contraseña*" field contains "*****". A red validation message box is present, stating: "Por favor, introduzca un nombre de usuario y clave correctos. Observe que ambos campos pueden ser sensibles a mayúsculas." Below the form, there are links for password recovery ("Restablecer!") and account creation ("Registrarse!").

Para quitar el formulario, añadimos en la condición anterior lo siguiente:

```
</ul>
{%
  if not request.user.is_authenticated and not "/accounts/login/" in request.get_full_path %}
<form class="navbar-form navbar-right" method="POST" action="{% url 'auth_login' %}">{% csrf_token %}
```

Vamos a probarlo:

The screenshot shows the same browser window and URL as the previous one. The validation message box is no longer present. The "Nombre de usuario*" field now contains "miguelgulu" and the "Contraseña*" field contains "*****". The rest of the page content, including the "Iniciar Sesión" button and the footer links, remains the same.

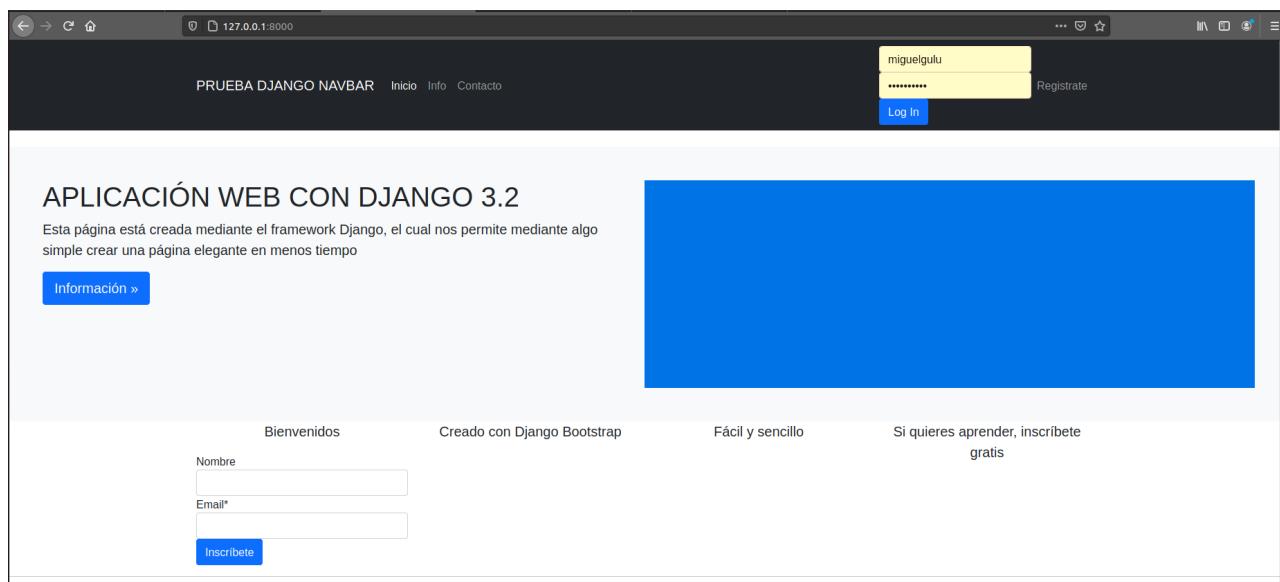
35. ESTILO: PERSONALIZAR CSS EN JUMBOTRON & Y NAVBAR

En este video vamos a modificar el estilo de nuestro contenedor principal en la página de inicio. Si en nuestro ‘base.html’ movemos todo el bloque de *container* fuera de la clase *container* este se abre y se extiende al estar fuera del bloque:

```

42  {% include "navbar.html" %}
43
44  {% block container-content %}
45  {% endblock %}
46
47  <main class="container">
48
49
50
51  </main>
52

```

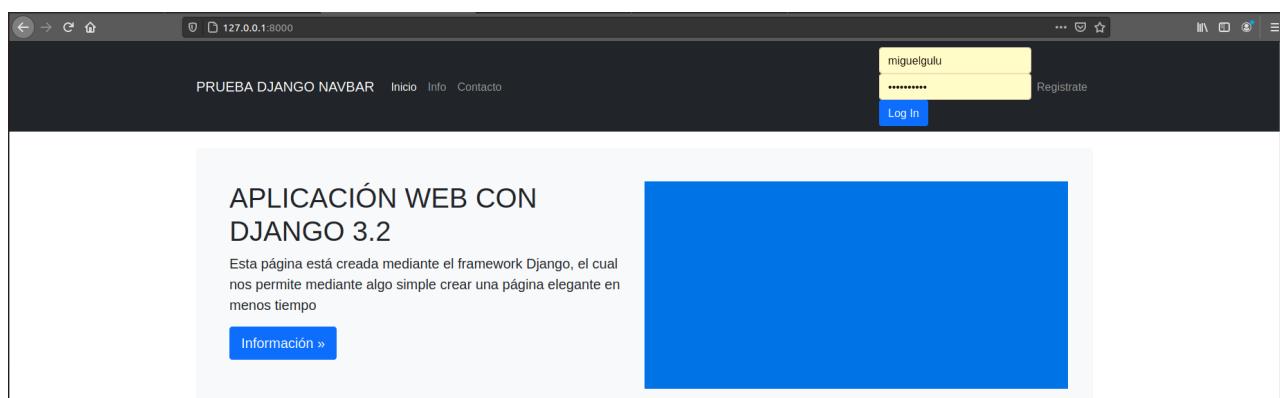


Vamos a mover el contenedor dentro del bloque que está en inicio:

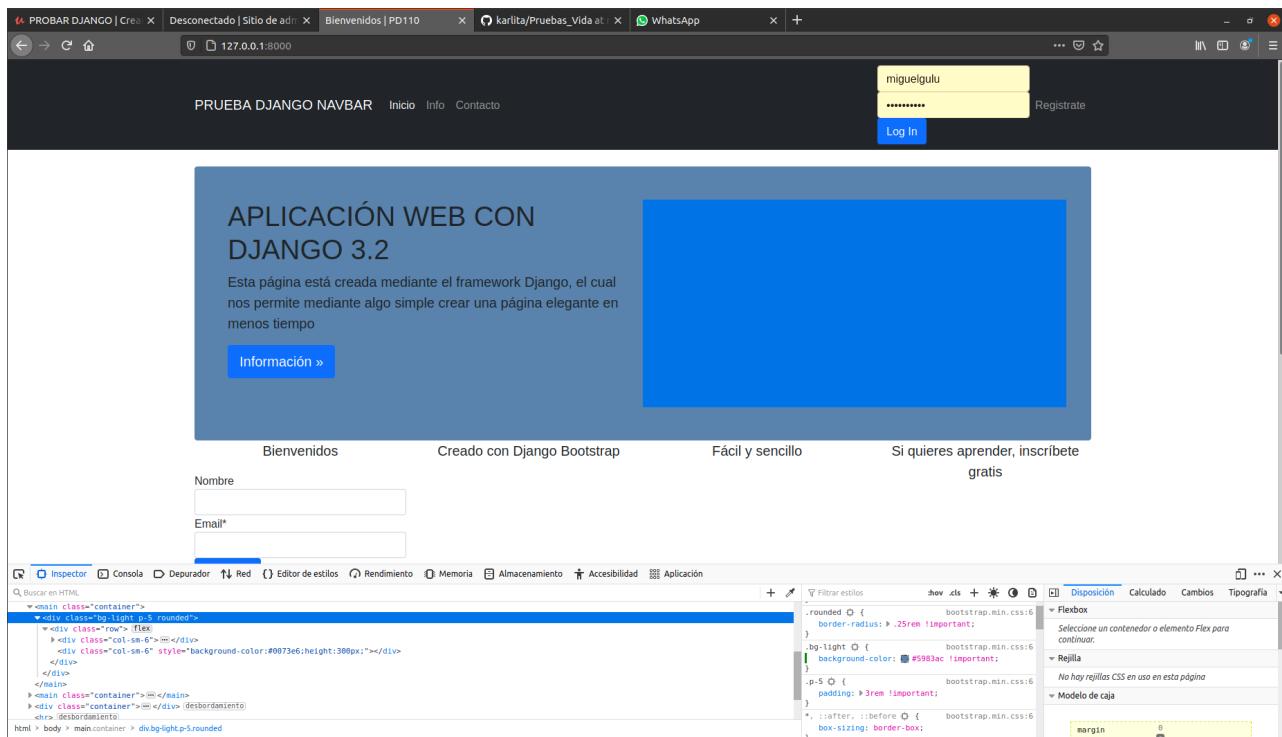
```

7  {% block container-content %}
8  <main class="container">
9
10 <div class="bg-light p-5 rounded">
11 <div class="row">
12 <div class="col-sm-6">
13   <h1>APLICACIÓN WEB CON DJANGO 3.2</h1>
14   <p>Esta página está creada mediante el framework Django, el cual nos permite mediante algo simple crear una página elegante en menos tiempo</p>
15   <a class="btn btn-lg btn-primary" href="/docs/5.0/components/navbar/" role="button">Información &raquo;</a>
16 </div>
17 <div class='col-sm-6' style='background-color:#0073e6; height:300px;'></div>
18 </div>
19 </div>
20
21 </main>
22 {% endblock %}
23

```



Y así lo tenemos todo mejor estructurado sin dejar cosas sueltas.



Inspeccionando elementos podemos probar distintas combinaciones de colores.

Esas pruebas no van a influir en la página. La única manera de cambiarlo es mediante el mismo código. Lo vamos a hacer por bloques. Para ello, vamos a nuestro archivo base y dentro del bloque de cabecera añadimos estilo:

```

15
16
17
18 <style>
19   {% block style %}{% endblock %}

```

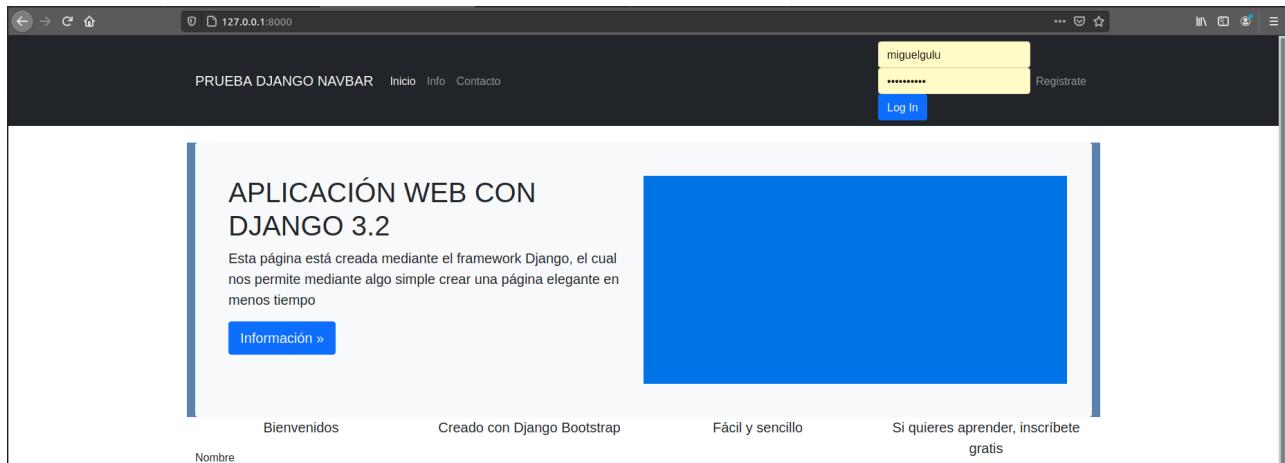
Y lo que contendrá estará en el inicio:

```

0
7   <style>
8   {% block style %}
9   .container-content {
10     background-color: #5983ac;
11   }
12   {% endblock %}
13 </style>
14

```

Recargamos la página...



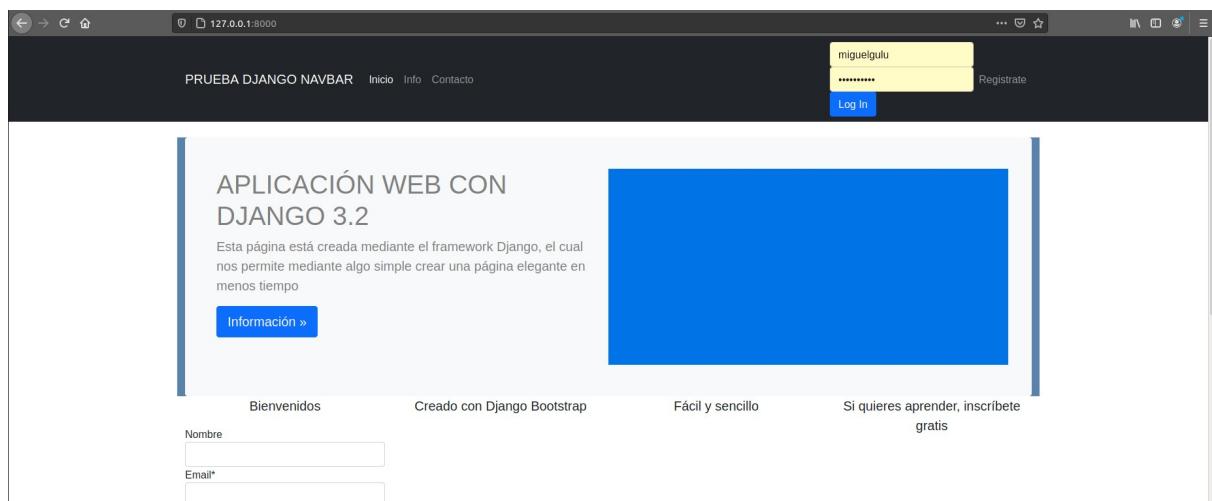
A nosotros nos sale distinto porque tenemos un contenedor dentro de otro y nuestra plantilla es distinta a la de karlita.

Vamos a cambiar a continuación a la fuente de dentro:

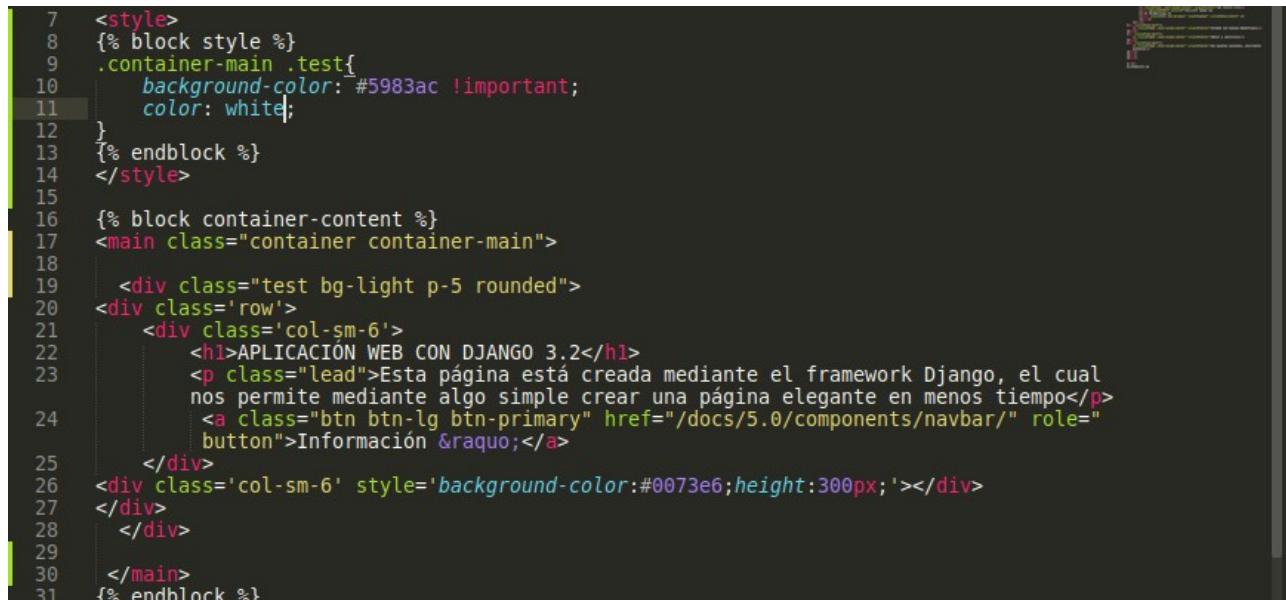
```

 7  <style>
 8  {% block style %}
 9  .container-main {
10    background-color: #5983ac !important;
11    color: grey;
12  }
13  {% endblock %}
14 </style>
15
16 {% block container-content %}
17 <main class="container container-main">
18
19   <div class="bg-light p-5 rounded">
20   <div class="row">
21     <div class="col-sm-6">
22       <h1>APLICACIÓN WEB CON DJANGO 3.2</h1>
23       <p class="lead">Esta página está creada mediante el framework Django, el cual
24         nos permite mediante algo simple crear una página elegante en menos tiempo</p>
25       <a class="btn btn-lg btn-primary" href="/docs/5.0/components/navbar/" role="button">Información &raquo;</a>
26     </div>
27   <div class='col-sm-6' style='background-color:#0073e6;height:300px;'></div>
28
29
30 </main>
31 {% endblock %}
32

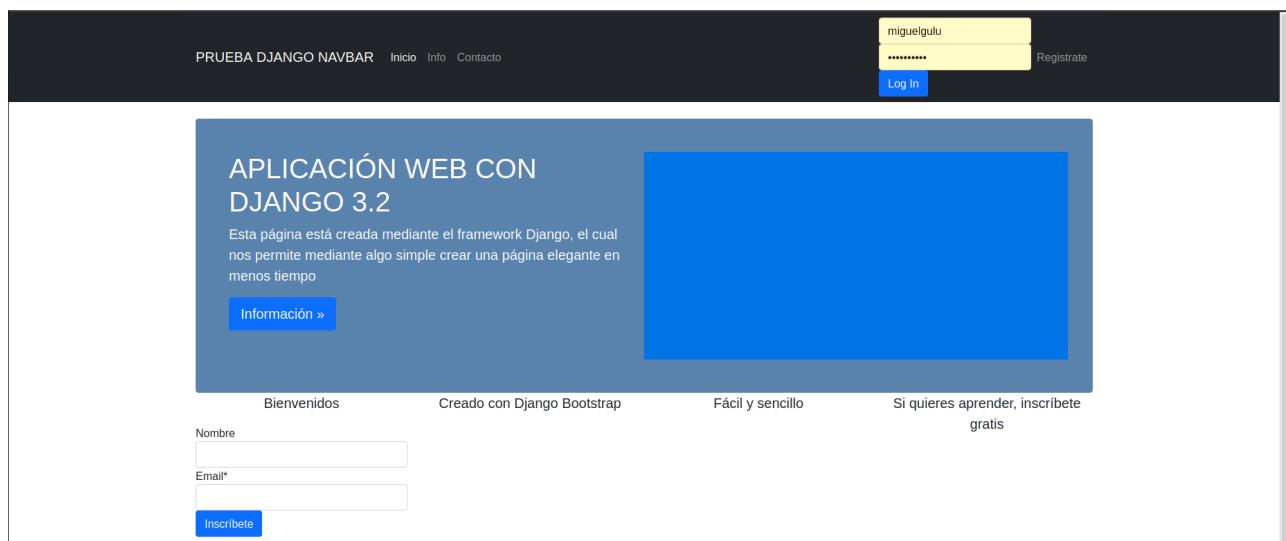
```



Voy a configurarlo por mi cuenta un poco más...

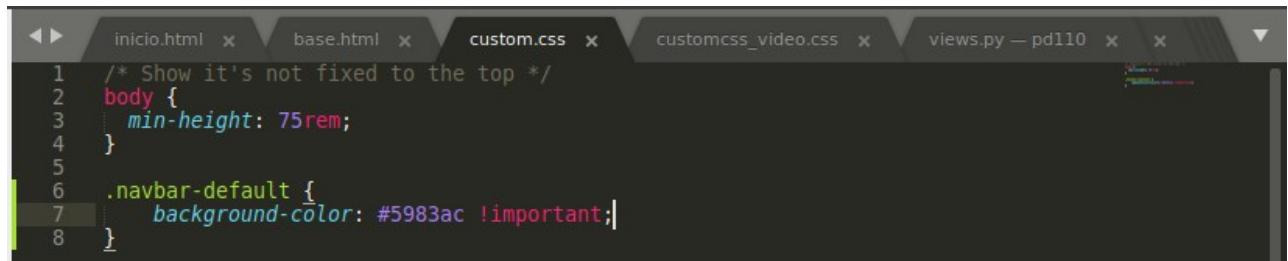


```
7 <style>
8 {%
9   block style %}
10 .container-main .test{
11   background-color: #5983ac !important;
12   color: white;
13 }
14 {%
15   endblock %}
16 </style>
17
18 {%
19   block container-content %}
20 <main class="container container-main">
21
22   <div class="test bg-light p-5 rounded">
23     <div class='row'>
24       <div class='col-sm-6'>
25         <h1>APLICACIÓN WEB CON DJANGO 3.2</h1>
26         <p class="lead">Esta página está creada mediante el framework Django, el cual
27           nos permite mediante algo simple crear una página elegante en menos tiempo</p>
28         <a class="btn btn-lg btn-primary" href="/docs/5.0/components/navbar/" role="button">Información &raquo;</a>
29       </div>
30     <div class='col-sm-6' style='background-color:#0073e6; height:300px;'></div>
31   </div>
32 </main>
33 {%
34   endblock %}
```



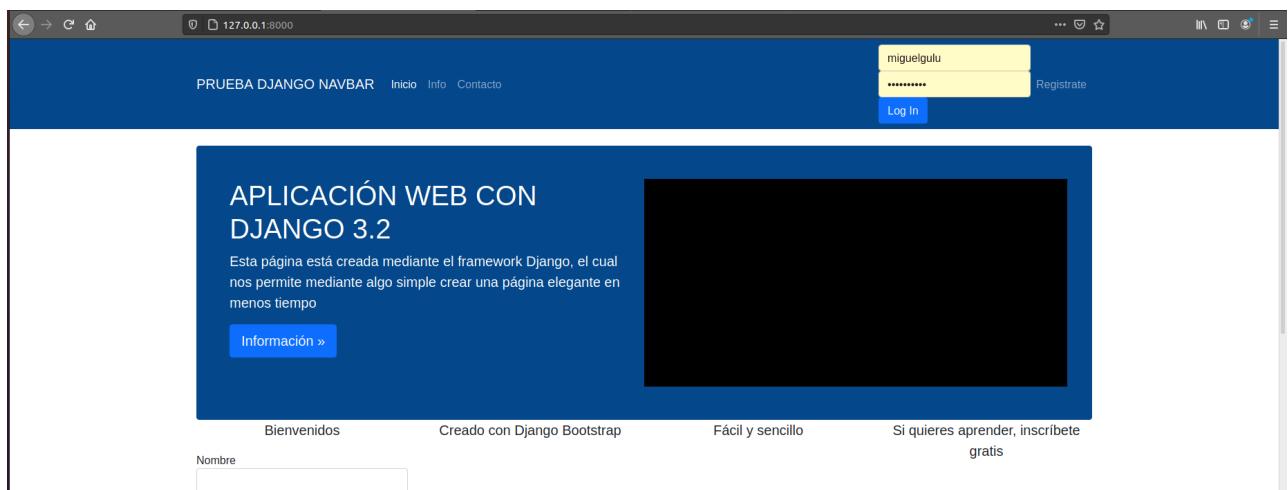
Listo!

Ahora toca modificar el color de la barra de navegación:



```
1 /* Show it's not fixed to the top */
2 body {
3     min-height: 75rem;
4 }
5
6 .navbar-default {
7     background-color: #5983ac !important;
8 }
```

Y comprobamos...



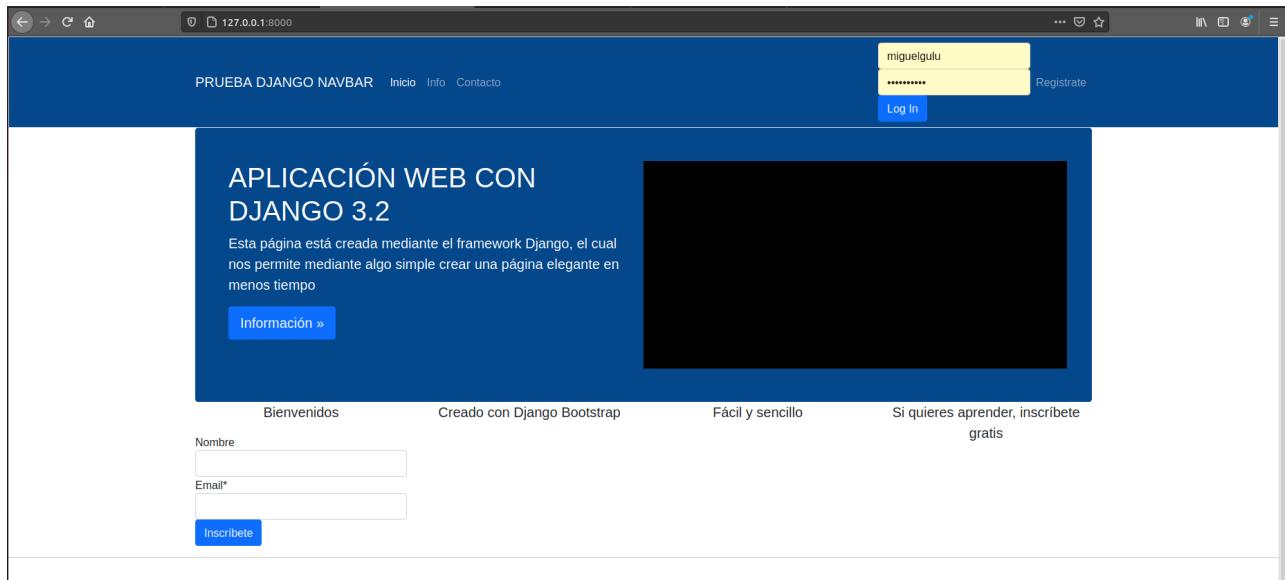
He retocado los colores un poco para que se vea mejor...

Ahora en el video nos muestran como quitar el espacio que hay entre el contenedor principal y el del navegador. Karlita lo hace cambiando la propiedad de margen a 0px pero, en nuestro caso, se hace borrando una clase la cual ya está definida en nuestra barra de navegación, la cual es 'mb-4':

```
<nav class="navbar navbar-expand-md navbar-dark bg-dark mb-4">
  <div class="container">
```

Y añadiremos un borde el cual no tenemos con una clase:

```
5 .border{
6   border-color: grey;
7 }
```



Ya tenemos nuestro borde gris en la barra de navegación.

Ya que hemos editado nuestros archivos de configuración, debemos subir los nuevos a la base de datos con `collectstatic`:

```
(venv) alunsiito@alunsiito:~/Documentos/LM/cursos/karlita/src$ python3 manage.py collectstatic
You have requested to collect static files at the destination
location as specified in your settings:

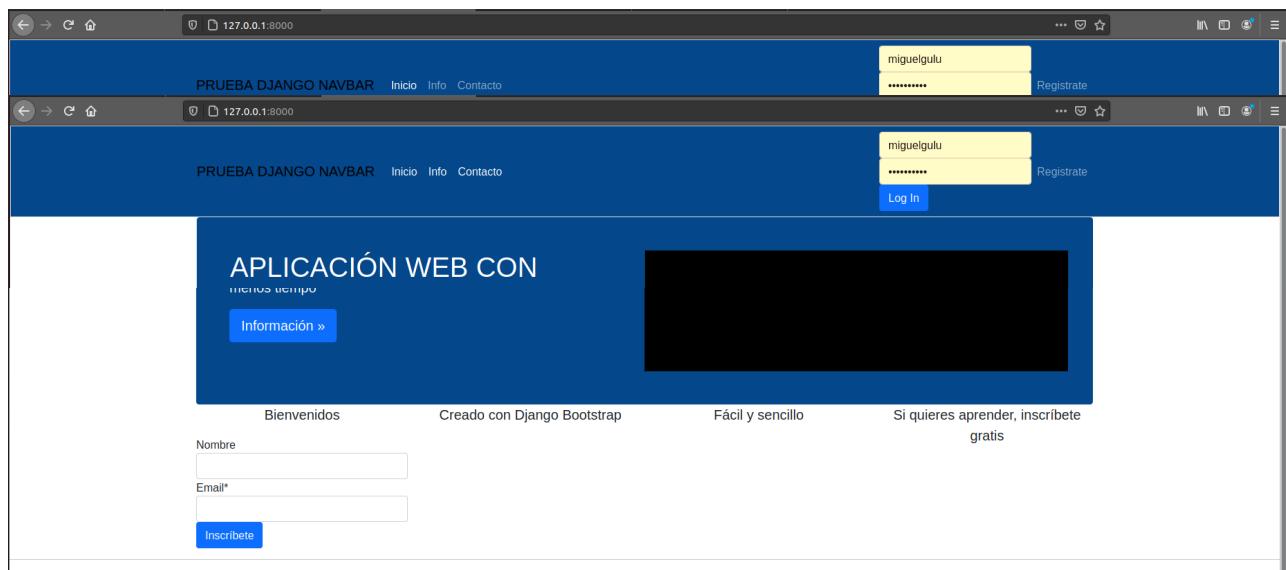
/home/alunsiito/Documentos/LM/cursos/karlita/static_env/static_root

This will overwrite existing files!
Are you sure you want to do this?

Type 'yes' to continue, or 'no' to cancel: yes
1 static file copied to '/home/alunsiito/Documentos/LM/cursos/karlita/static_env
/static_root', 131 unmodified.
(venv) alunsiito@alunsiito:~/Documentos/LM/cursos/karlita/src$
```

Una vez hecho, vamos a seguir configurando nuestra barra. Vamos a poner la letra grande de otro color igual que hemos hecho al configurar el estilo de lo anterior:

```
9 .navbar-brand {
10   color: black;
11 }
```

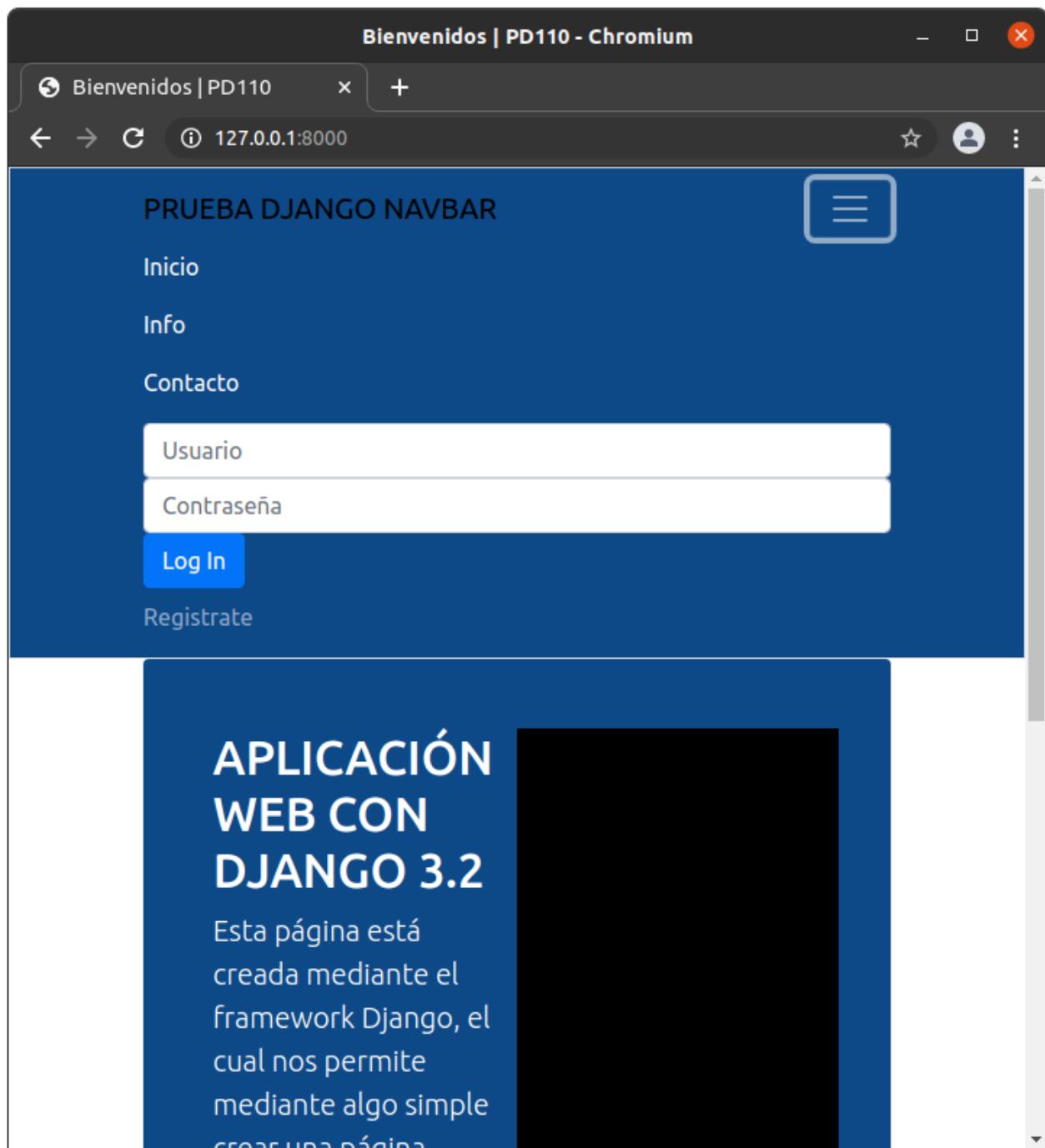


Si queremos hacer que se nos remarque el texto cuando pasamos por encima el ratón:

```
10  .marcado:hover {  
11    color: red;  
12 }
```

He creado una clase llamado ‘marcado’ y esta tiene la propiedad de que cambia el color cuando pasa por encima el ratón.

Si encogemos la ventana la barra de navegación se comprimirá de forma que, al dar en el botón que vemos, se mostrará lo que tenía:



Y si no queremos hacer ningún cambio más hacemos el correspondiente collectstatic:

```
^C(venv) alunsiito@alunsiito:~/Documentos/LM/cursos/karlita/src$ python3 manage.py collectstatic

You have requested to collect static files at the destination
location as specified in your settings:

/home/alunsiito/Documentos/LM/cursos/karlita/static_env/static_root

This will overwrite existing files!
Are you sure you want to do this?

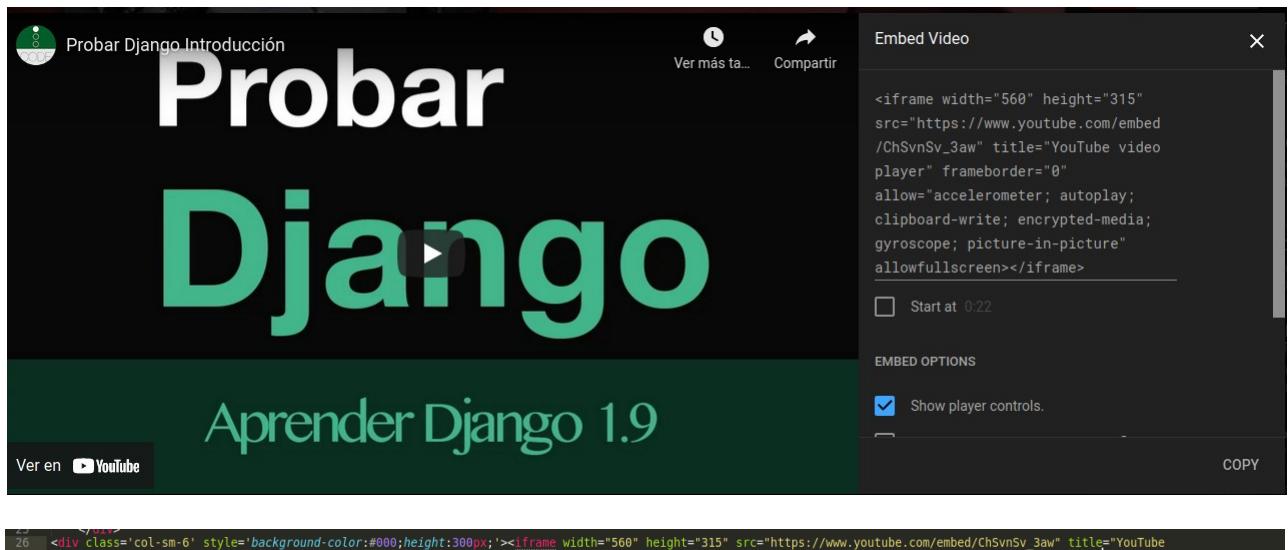
Type 'yes' to continue, or 'no' to cancel: yes

2 static files copied to '/home/alunsiito/Documentos/LM/cursos/karlita/static_en
v/static_root', 130 unmodified.
(venv) alunsiito@alunsiito:~/Documentos/LM/cursos/karlita/src$ 
```

36. AÑADIR VIDEO DE YOUTUBE A JUMBOTRON

En este video vamos a añadir un video de Youtube al recuadro que tenemos que, en mi caso es el ‘container-content’ y en el suyo es Jumbotron.

Para ello, nos vamos al video de Youtube que queramos publicar y en la opción de ‘Compartir’ elegimos ‘Embed’ y copiamos todo el código y lo añadimos dentro del tag de ‘div’:



Y tendremos que cambiar el tamaño para que se ajuste al video, de forma que el resultado es este:



Si borramos el atributo *style*, se quedará perfectamente ajustado:

APLICACIÓN WEB CON DJANGO 3.2

Esta página está creada mediante el framework Django, el cual nos permite mediante algo simple crear una página elegante en menos tiempo

[Información »](#)



Ahora vamos a probar a añadir una imagen. Para ello, buscamos una cualquiera y obtenemos su enlace. Nos vamos a la página de inicio y quitamos el tag de frame y añadimos uno de imagen. Queda de la siguiente manera:

```
25 |     </div>
26 |     <div class='col-sm-6'>
27 |     </div>
28 |   </div>
```

APLICACIÓN WEB CON DJANGO 3.2

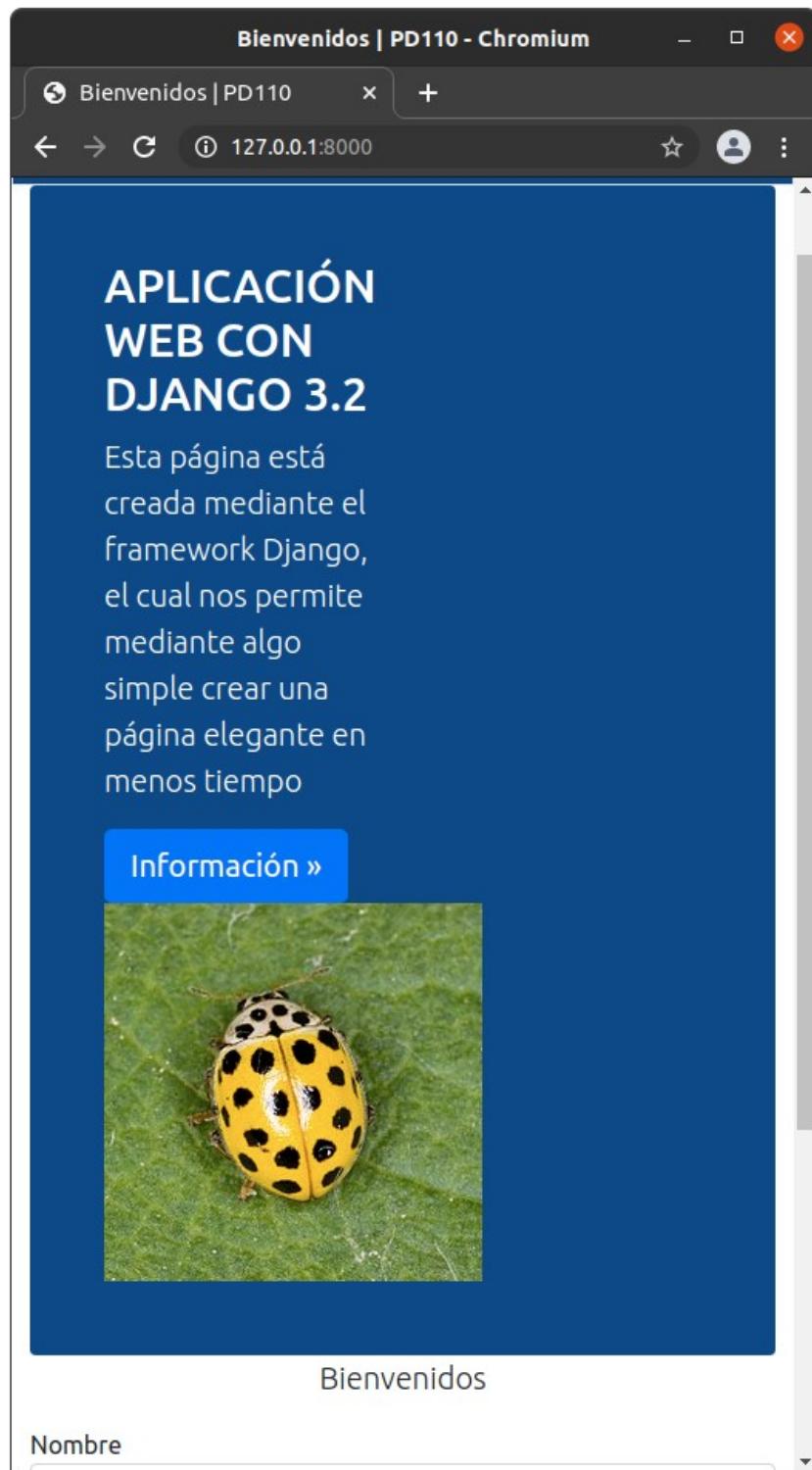
Esta página está creada mediante el framework Django, el cual nos permite mediante algo simple crear una página elegante en menos tiempo

[Información »](#)

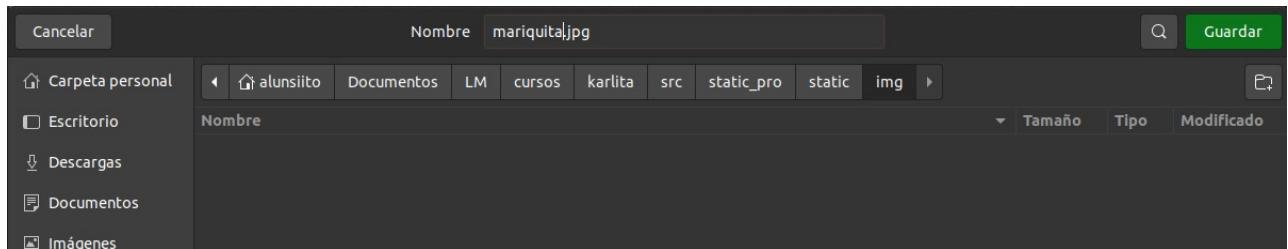


Y si queremos que la imagen sea responsive añadimos que sea de clase ‘*img-responsive*’:

```
25 |     </div>
26 |     <div class='col-sm-6'></div>
28 |   </div>
```



Esta es una manera de añadir una imagen, pero no es recomendable hacerlo directamente desde la url, ya que, si esta sufre un cambio, no se vería. La manera más efectiva es descargarla y añadirla a la carpeta *img* en los archivos estáticos.



Y añadimos la ruta:

```
25 |     </div>
26 | <div class='col-sm-6'></div>
27 | </div>
```

Comprobamos que funciona perfectamente:



A karlita le da un error porque no tiene cargado static en el inicio, yo ya lo tenía cargado de antes:

```
inicio.html x base.html x custom.css x bootstrap.min.css x navbar.htm
1  {% extends "base.html" %}
2  {% load static %}
3  {% load crispy_forms_tags %}
```

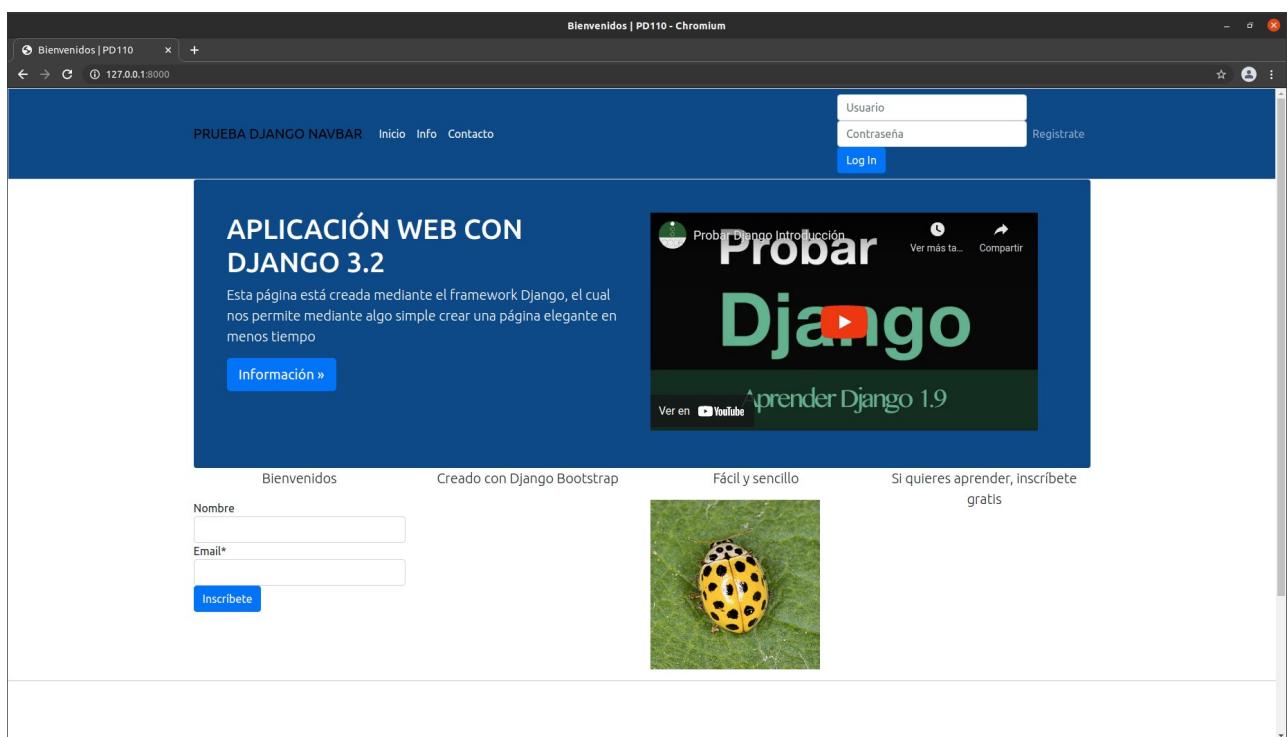
Vamos a volver a dejar el video para que quede mejor y damos por concluido el capítulo.

37. FONTAWESOME

Aquí lo que vamos a hacer es añadir la foto que guardamos en el vídeo anterior pero en la parte de abajo del contenedor.

```
52 <div class="csc-sim-3">
53   <p class="lead text-align-center" align="center">Fácil y sencillo</p>
```

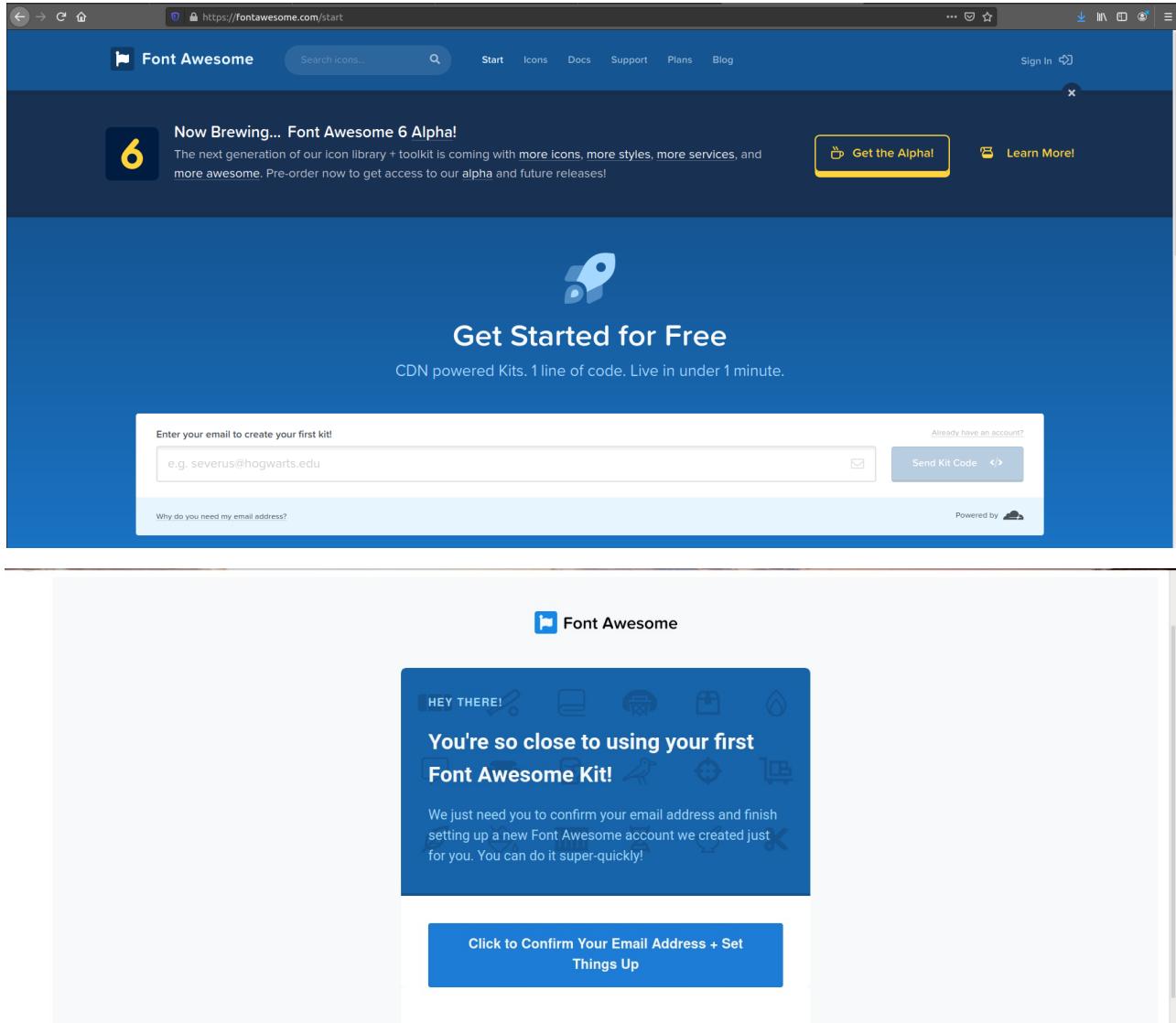
Y así quedaría:



Volvemos a hacer collectstatic para subir la imagen...

```
^C(venv) alunsiito@alunsiito:~/Documentos/LM/cursos/karlita/src$ python3 manage.py collectstatic
You have requested to collect static files at the destination location as specified in your settings:
/home/alunsiito/Documentos/LM/cursos/karlita/static_env/static_root
This will overwrite existing files!
Are you sure you want to do this?
Type 'yes' to continue, or 'no' to cancel: yes
1 static file copied to '/home/alunsiito/Documentos/LM/cursos/karlita/static_env/static_root', 132 unmodified.
(venv) alunsiito@alunsiito:~/Documentos/LM/cursos/karlita/src$
```

A partir de aquí vamos a trabajar, como dice el título, con Font Awesome. Es una colección de iconos ya dispuesto en CSS para tener una página más dinámica. Para empezar a usarlo basta con ir a la página principal y a 'Start Free'. Nos saldrá para que nos envíe un kit a nuestro correo para poder usar recursos gratis.



Nos llegará este correo y tendremos que confirmar la dirección.

First, choose a password for your new Font Awesome account.

New Password

🔒

Make it a good one!

Confirm New Password

🔒

Make it match the good one!

Set Password & Continue →

Nos enviará un enlace para elegir nuestra contraseña...

Let's Get to Know You Better

Mind sharing a few bits about yourself?

What's your first name?

Miguel

How about your last name?

Guerrero

You first used Font Awesome in...

Haven't used it yet

Got a favorite Font Awesome icon?

e.g. fa-rocket

All set. Let's go! →

[No thanks. Let's skip this step for now.](#)

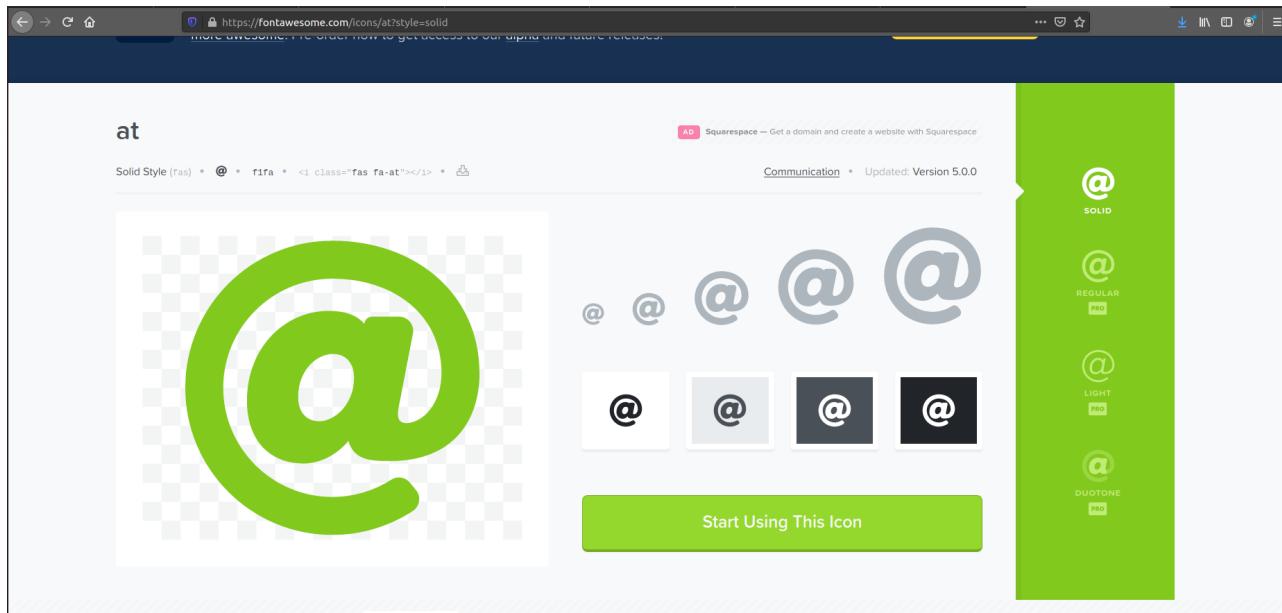
Rellenamos con nuestros datos... (La última casilla es opcional)

Y listo!

Basta con ir a nuestro perfil y en nuestros Kit's tendremos el código que tendremos que añadir a head_css:

```
<script src="https://kit.fontawesome.com/2ce0fc84db.js" crossorigin="anonymous"></script>
```

Nosotros vamos a usar solamente los iconos que podemos añadir de forma gratuita que vienen incluidos en la página principal. Para ello vamos a icons y elegimos uno que esté marcado en oscuro. El resto no se podrá usar ya que son premium.



Este es el que voy a escoger yo.

Si clickamos en '**Start using this icon**' se nos abrirá una ventana que contiene el código que tendríamos que añadir para desplegar el ícono en nuestra página.

```
49  <div class="col-sm-3">
50    <p class="lead text-align-center" align="center">Creado con Django Bootstrap @
51      class="fas fa-at"></i></p>
52  </div>
```

Y este es el resultado:

Bienvenidos	Creado con Django Bootstrap @	Fácil y sencillo	Siquieres aprender, inscríbete gratis
<input type="text" value="Nombre"/>			

Tenemos ya nuestro ícono añadido!

Vamos a hacerlo más grande:

```
<div class="col-sm-3">
  <p class="lead text-align-center" align="center">Creado con Django Bootstrap <i
    class="fas fa-at fa-5x"></i></i></p>
</div>
```

Y se ve así:

Bienvenidos

Nombre

Email*

Creado con Django Bootstrap

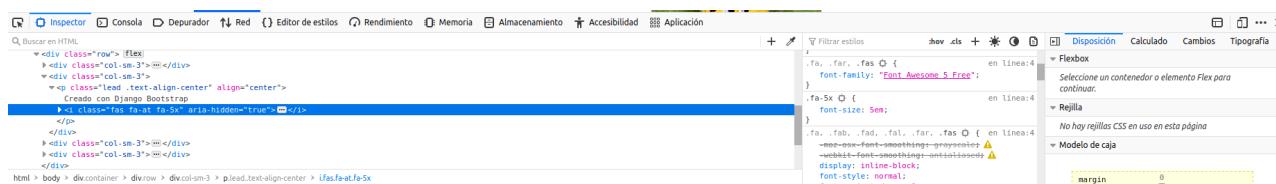


Fácil y sencillo

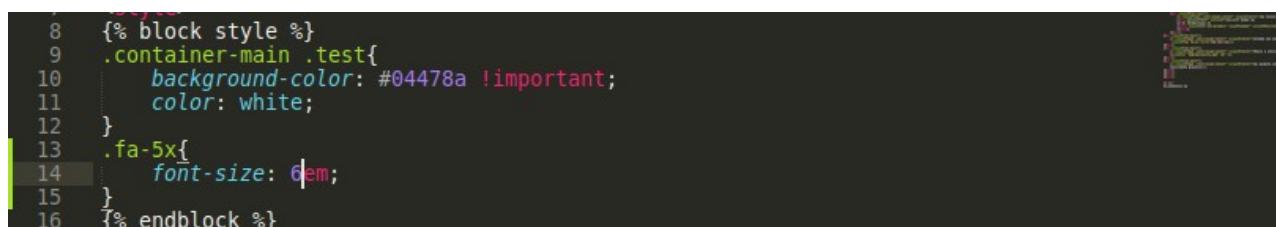


Si quieres aprender, inscríbete gratis

Si inspeccionamos el elemento podemos ver como aparece como tamaño de fuente. Esto se debe a que el ícono no se considera como imagen, sino como texto.



Vamos a configurarlo en nuestro *block style*:



```

8  {% block style %}
9  .container-main .test{
10    background-color: #04478a !important;
11    color: white;
12  }
13  .fa-5x{
14    font-size: 6em;
15  }
16  {% endblock %}

```

Bienvenidos

Nombre

Email*

Creado con Django Bootstrap

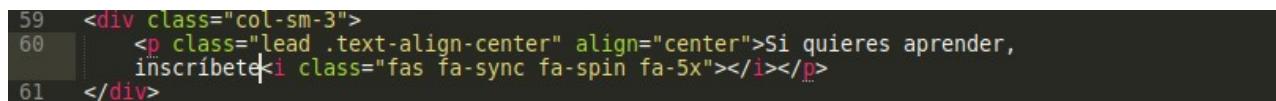


Fácil y sencillo



Si quieres aprender, inscríbete gratis

También podemos añadir iconos animados siguiendo el mismo procedimiento:



```

59  <div class="col-sm-3">
60    <p class="lead .text-align-center" align="center">Si quieres aprender,
61      inscríbete<i class="fas fa-sync fa-spin fa-5x"></i></p>
62  </div>

```

Bienvenidos

Nombre

Email*

Inscríbete

Creado con Django Bootstrap



Fácil y sencillo



Si quieres aprender, inscríbete



El ícono animado tiene movimiento gracias a la clase ‘fa-spin’. Podemos añadírselo al otro ícono:

Bienvenidos Creado con Django Bootstrap Fácil y sencillo Si quieras aprender, inscríbete

Nombre

Email*







Además podemos añadir iconos stackeados unos encima de otros:

```
60 <p class="lead text-align-center" align="center">Si quieres aprender,  
61 inscríbete<span class="fa-stack fa-2x">  
62 <i class="fas fa-square fa-stack-2x"></i>  
63 <i class="fab fa-twitter fa-stack-1x fa-inverse"></i>  
</span></p>
```

Bienvenidos Creado con Django Bootstrap Fácil y sencillo Si quieras aprender, inscríbete

Nombre

Email*







Lo ponemos a mayor tamaño..

Fácil y sencillo Si quieres aprender, inscríbete





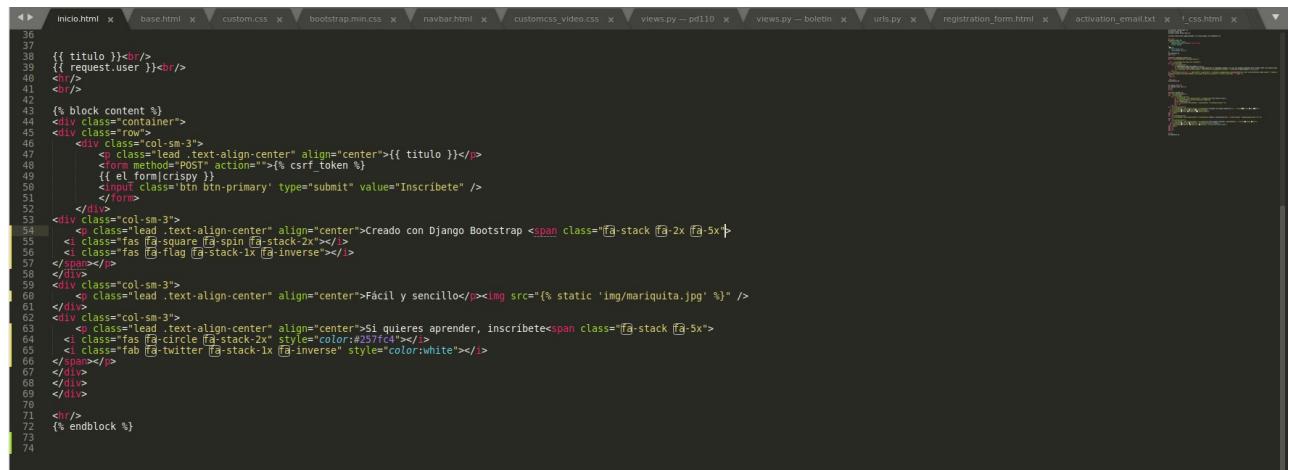
Podemos superponer otra imagen que no sea el cuadrado, como por ejemplo:

Fácil y sencillo Si quieres aprender, inscríbete





Y podemos modificar parámetros como los colores:

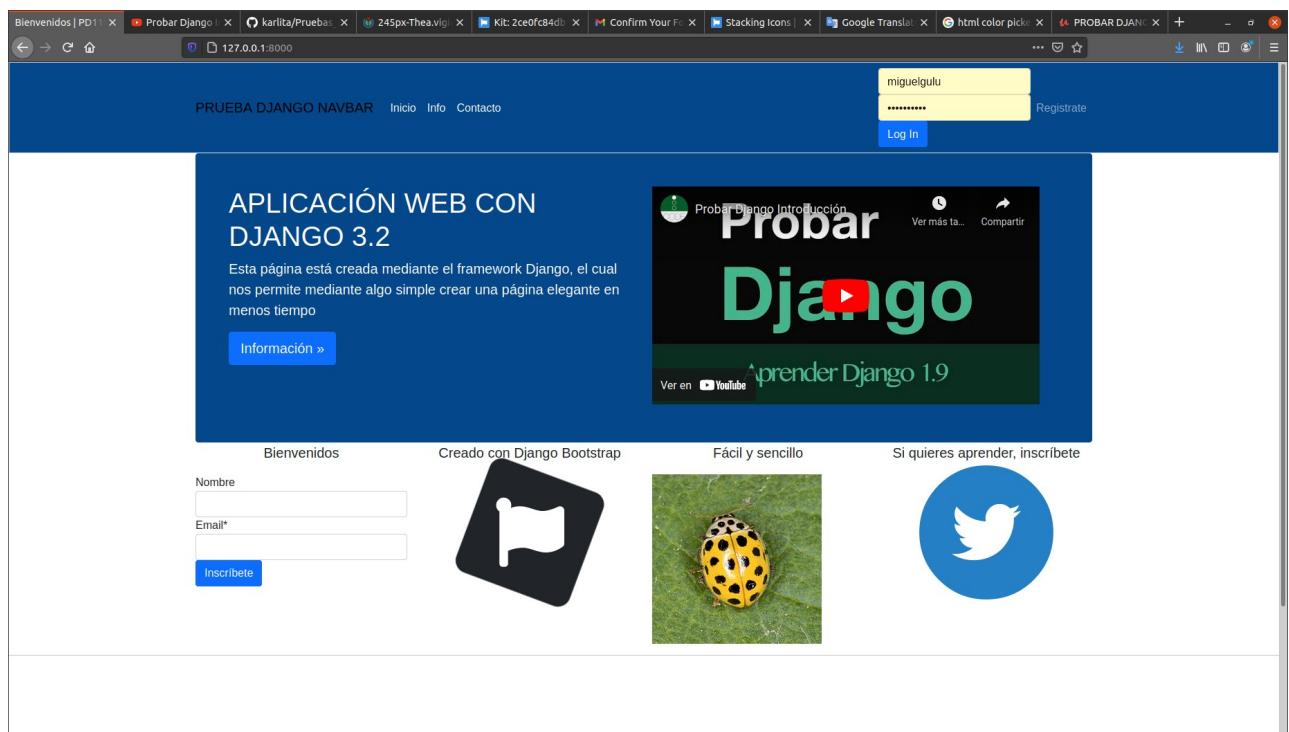


```

36 {{ titulo }}<br/>
37 {{ request.user }}<br/>
38 <br/>
39 <div class="block">
40   <div class="block content">
41     <div class="container">
42       <div class="row">
43         <div class="col-sm-3">
44           <p class="lead text-align-center" align="center">{{ titulo }}</p>
45           <form method="POST" action="{% csrf_token %}">
46             {{ el_formcrispy }}
47             <input class="btn btn-primary" type="submit" value="Inscríbete" />
48           </form>
49         </div>
50       </div>
51     </div>
52   </div>
53   <div class="col-sm-3">
54     <p class="lead text-align-center" align="center">Creado con Django Bootstrap <span class="fa-stack fa-2x fa-5x">
55       <i class="fas fa-square fa-spin fa-stack-2x"></i>
56       <i class="fas fa-flag fa-stack-1x fa-inverse"></i>
57     </span></p>
58   </div>
59   <div class="col-sm-3">
60     <p class="lead text-align-center" align="center">Fácil y sencillo</p>
61   </div>
62   <div class="col-sm-3">
63     <p class="lead text-align-center" align="center">Si quieres aprender, inscríbete<span class="fa-stack fa-5x">
64       <i class="fas fa-circle fa-stack-2x" style="color:#257742"></i>
65       <i class="fab fa-twitter fa-stack-1x fa-inverse" style="color:white"></i>
66     </span></p>
67   </div>
68 </div>
69 </div>
70 <br/>
71 {% endblock %}
72
73
74

```

Y este es el resultado final:



38. CONTENIDO PARA USUARIOS AUTENTICADOS

Para empezar este capítulo, nos vamos a dedicar un poco a modificar nuestra página, empezando por quitar el formulario que tenemos por debajo de Bienvenidos. Entonces ponemos un condicional para que solo salga si estamos sin autenticar:

```

46     <div class="col-sm-3">
47         <p class="lead text-align-center" align="center">{{ titulo }}</p>
48         {% if not user.is_authenticated %}
49             <form method="POST" action="">{{ csrf_token }}
50                 {{ el_form|crispy }}
51             <input class='btn btn-primary' type="submit" value="Inscríbete" />
52         </form>
53     {% endif %}
54 </div>
```

The screenshot shows a Django application's homepage. At the top, there is a navigation bar with links for 'PRUEBA DJANGO NAVBAR', 'Inicio', 'Info', 'Contacto', and 'Salir'. The main content area has a dark blue background. On the left, there is a 'Bienvenido miguelgulu' message and a 'Creado con Django Bootstrap' badge. In the center, there is a large 'Probar Django' heading with a 'Aprender Django 1.9' button below it. On the right, there is a 'Fácil y sencillo' message and a 'Si quieres aprender, inscríbete' button. At the bottom, there are social media icons for Twitter and a yellow ladybug image.

Y como estamos autenticados con nuestro usuario, el formulario no aparece.

The screenshot shows the same Django application after logging in. The 'Bienvenido miguelgulu' message has changed to 'Bienvenido pruebesita'. The 'Creado con Django Bootstrap' badge is still present. The rest of the page content remains the same, including the 'Probar Django' heading and the social media icons.

Si iniciamos sesión con un usuario femenino sigue apareciendo 'Bienvenido' en vez de tener en cuenta el género. Vamos a ajustar eso:

```
if request.user.is_authenticated:  
    titulo = "Bienvenid@ %s" %(request.user)  
    form = RegModelForm(request.POST or None)
```

En nuestra vista, al título le añadiremos el @ para que se decida el género automáticamente...

Bienvenid@ pruebesita Crea

A continuación vamos a añadir un icono donde debería estar el formulario:

```
51     <input class='btn btn-primary' type="submit" value="Inscríbete" />  
52 </form>  
53 {% else %}  
54     <i class="fas fa-peace"></i>  
55 {% endif %}
```

Y lo añadimos como *else* para el caso de que estemos autenticados.

Bienvenid@ pruebesita



Ahi estaría. Vamos a ajustarlo un poco...

Bienvenid@ pruebesita Creado con Django Bootstrap



Mejor!

Vamos a añadir un título nuevo en el caso de que estemos *logueados*:

```
    <p class="lead text-align-center" align="center">{{ titulo }}<br>
    </br></p>
    <i class="fas fa-peace fa-5x col-12"></i>
```

El problema de esto es que se verá el título 2 veces...

Bienvenid@ pruebesita

Creado con Django Bootstrap

Bienvenid@ pruebesita



Para solucionarlo:

```
46      <div class="col-sm-3">
47          {% if not user.is_authenticated %}
48              <p class="lead text-align-center" align="center">Suscríbete</p>
49              <form method="POST" action="">{% csrf_token %}
50                  {{ el_form|crispy }}
51                  <input class='btn btn-primary' type="submit" value="Inscríbete" />
52              </form>
53          {% else %}
54              <p class="lead text-align-center" align="center">{{ titulo }}<br>
55                  </br></p>
56                  <i class="fas fa-peace fa-5x col-12"></i>
57          {% endif %}
```

Y quedaría así:

Bienvenid@ pruebesita

Creado con Django Bootstrap

Fácil y sencillo

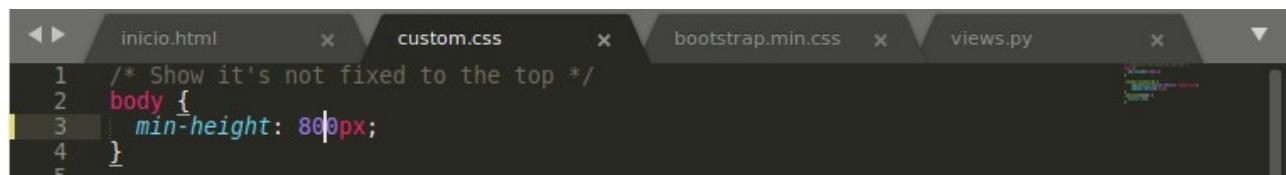
Siquieres aprender, inscríbete



Y si cerramos sesión, nos saldrá el título de suscribirse:



A continuación quitaremos toda la parte de blanco que sobra de la página que no hemos usado...



Modificamos el tamaño que, a nosotros de manera predeterminada nos viene en 70rem, el cual es otro sistema de medida. Podemos cambiarlo a px de píxeles.

Como hemos modificado los archivos estáticos, toca otro *collectstatic*.

```
^C(venv) alunsiito@alunsiito:~/Documentos/LM/cursos/karlita/src$ python3 manage.py collectstatic
You have requested to collect static files at the destination location as specified in your settings:
/home/alunsiito/Documentos/LM/cursos/karlita/static_env/static_root
This will overwrite existing files!
Are you sure you want to do this?
Type 'yes' to continue, or 'no' to cancel: yes
2 static files copied to '/home/alunsiito/Documentos/LM/cursos/karlita/static_env/static_root', 131 unmodified.
(venv) alunsiito@alunsiito:~/Documentos/LM/cursos/karlita/src$
```

Finalmente, vamos a darle una vista a nuestro 'about' que teníamos pendiente.

TemplateDoesNotExist at /about/
about.html

```

Request Method: GET
Request URL: http://127.0.0.1:8000/about/
Django Version: 3.2
Exception Type: TemplateDoesNotExist
Exception Value: about.html
Exception Location: /home/alunsito/Documentos/LM/cursos/karlita/venv/lib/python3.8/site-packages/django/template/loader.py, line 19, in get_template
Python Executable: /home/alunsito/Documentos/LM/cursos/karlita/venv/bin/python3
Python Version: 3.8.5
Python Path: ['/home/alunsito/Documentos/LM/cursos/karlita/src',
 '/usr/local/lib/python3.8.zip',
 '/usr/local/lib/python3.8',
 '/usr/local/lib/python3.8/lib-dynload',
 '/home/alunsito/Documentos/LM/cursos/karlita/venv/lib/python3.8/site-packages']
Server time: Sun, 25 Apr 2021 10:39:00 +0000

```

Template-loader postmortem

Django tried loading these templates, in this order:

Using engine django:

- django.template.loaders.filesystem.Loader: /home/alunsito/Documentos/LM/cursos/karlita/src/templates/about.html (Source does not exist)

Pues crearemos nuestra plantilla llamada '*about.html*' en la carpeta de *templates*, la cual se leerá automáticamente y se verá en cuanto escribamos la dirección:

```

File Edit Selection Find View Goto Tools Project Preferences Help
FOL inicio.html x custom.css x about.html x bootstrap.min.css x views.py x
1  {% extends "base.html" %} 
2
3  {% block content %} 
4      <div class="row">
5          <div class="col-sm-6 col-sm-offset-3">
6              <h2 class="text-align-center"><strong>¿Quién soy?</strong></h2>
7              <p>Soy Miguel Guerrero Luna y estoy aprendiendo a usar el Framework Django.
8                  Quieres saber como aprendí? Pues aquí tienes todo lo que necesitas! </p>
9              <p>Mucho texto mucho texto mucho texto mucho texto</p>
10             <p>Mucho texto mucho texto mucho texto mucho texto</p>
11             <p>Mucho texto mucho texto mucho texto mucho texto</p>
12             <p>Mucho texto mucho texto mucho texto mucho texto</p>
13             <p>Mucho texto mucho texto mucho texto mucho texto</p>
14             <p>Mucho texto mucho texto mucho texto mucho texto</p>
15             <p>Mucho texto mucho texto mucho texto mucho texto</p>
16         </div>
17     </div>
18
19
20  {% endblock %} 

```

127.0.0.1:8000/about/

PRUEBA DJANGO NAVBAR Inicio Info Contacto Salir

¿Quién soy?

Soy Miguel Guerrero Luna y estoy aprendiendo a usar el Framework Django. Quieres saber como aprendí? Pues aquí tienes todo lo que necesitas!

Mucho texto mucho texto mucho texto mucho texto
Mucho texto mucho texto mucho texto mucho texto

Lo siguiente que queremos añadir a nuestra página es un QuerySet el cual hará que se muestre en la página todos los usuarios registrados. Para ello nos vamos a nuestros *views* y al final añadimos un nuevo contexto si estamos autenticados y somos *staff*:

```
if request.user.is_authenticated() and request.user.is_staff():
    context = {
        "queryset": ['hola', 'adios'],
    }

    return render (request, "inicio.html", context)
```

(Hay que quitar los paréntesis ya que corresponde a un valor booleano, dejando esto así da error)

Y luego nos vamos a nuestro inicio y añadimos un condicional para que muestre el QuerySet si es un usuario staff:

```
22  {% if request.user.is_staff %}
23  {{ queryset }}
24  {% else %}
25
26 <main class="container container-main">
27
28     <div class="test bg-light p-5 rounded">
29         <div class="row">
30             <div class="col-6">
31                 <h1>APLICACIÓN WEB CON DJANGO 3.2</h1>
32                 <p class="lead">Esta página está creada mediante el framework Django, el cual nos permite mediante algo simple crear una página elegante en menos tiempo</p>
33                 <a class="btn btn-lg btn-primary" href="/docs/5.0/components/navbar/" role="button">Información &raquo;</a>
34             </div>
35         <div class="col-sm-6"><iframe width="560" height="315" src="https://www.youtube.com/embed/ChSvnSv_3aw" title="YouTube video player" frameborder="0" allow="accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture" allowfullscreen></iframe>
36     </div>
37     </div>
38
39 </main>
40 {% endif %}
41 {% endblock %}
```

Y vamos a quitar todo el ‘block content’ si estamos autenticados:

```
49  {% block content %}
50  {% if not request.user.is_staff %}
51  <div class="container">
52      <div class="row">
53          <div class="col-sm-3">
54              {% if not user.is_authenticated %}
55                  <p class="lead .text-align-center" align="center">Suscríbete</p>
56                  <form method="POST" action="{% csrf_token %}">
57                      {{ el form|crispy }}
58                      <input class='btn btn-primary' type="submit" value="Inscríbete" />
59                  </form>
60              {% else %}
61                  <p class="lead .text-align-center" align="center">{{ titulo }}</p>
62                  <i class="fas fa-peace fa-5x col-12" style="font-size: 10em;"></i>
63              {% endif %}
64          </div>
65      <div class="col-sm-3">
66          <p class="lead .text-align-center" align="center">Creado con Django Bootstrap </p>
67          <span class="fa-stack fa-2x fa-5x col-12">
68              <i class="fas fa-square fa-spin fa-stack-2x"></i>
69              <i class="fas fa-flag fa-stack-1x fa-inverse"></i>
70          </span>
71      </div>
72      <div class="col-sm-3">
73          <p class="lead .text-align-center" align="center">Fácil y sencillo</p>
75      </div>
76      <div class="col-sm-3">
77          <p class="lead .text-align_center" align="center">Si quieres aprender, inscríbete</p>
78          <span class="fa-stack fa-5x col-12">
79              <i class="fas fa-circle fa-stack-2x" style="color:#257fc4"></i>
80              <i class="fab fa-twitter fa-stack-1x fa-inverse" style="color:white"></i>
81          </span>
82      </div>
83  </div>
84
85  <hr>
86  {% endif %}
87  {% endblock %}
```

Guardamos y comprobamos...



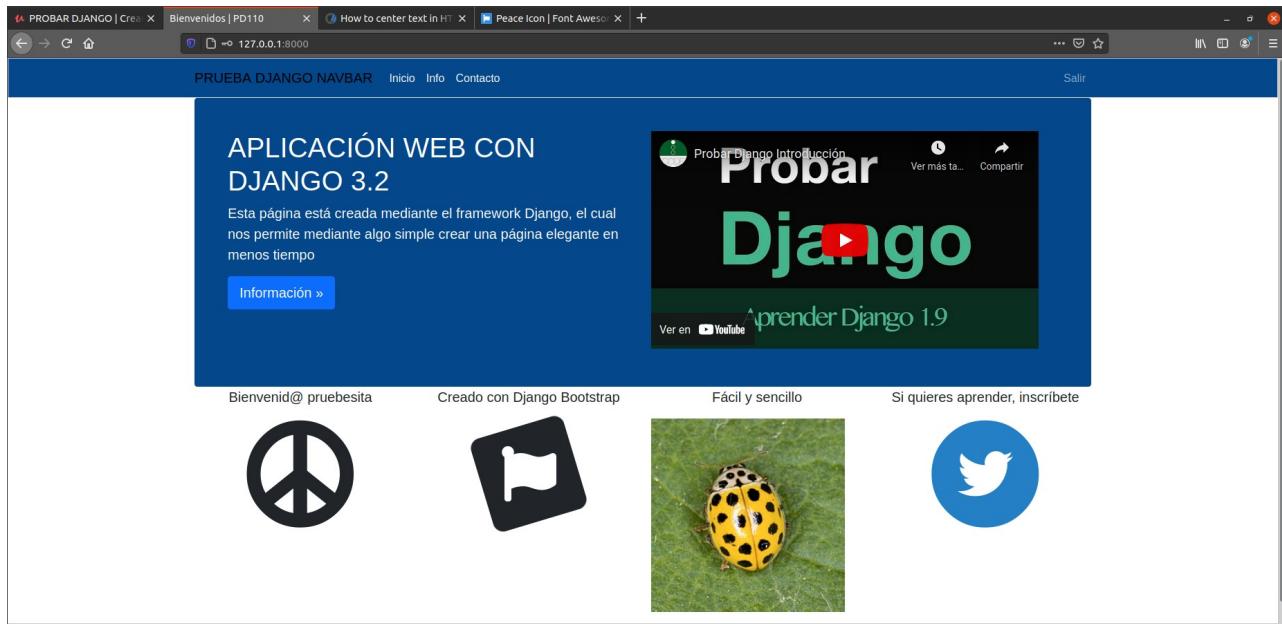
Y listo! Así sería la vista previa de nuestro QuerySet, al cual le dedicaremos el próximo vídeo.

39. INTRODUCCIÓN BÁSICA A LOS QUERYSETS

En este vídeo vamos a aprender a hacer QuerySets, osea, recuperar datos guardados en la base de datos a través del modelo.

Vamos a continuar por donde dejamos el último vídeo.

Si iniciamos sesión con un usuario que no es *staff* como ‘pruebesita’, veremos que no nos sale el Queryset:



Para recorrer esta lista de datos usaremos un FOR.

Debemos tener nuestro modelo importado con la clase que creamos (Registrado). Empezemos por mostrar todos los objetos:

```
if request.user.is_authenticated and request.user.is_staff:
    print ("Registrado.objects.all()")
    context = {
        "queryset": ['hola', 'adios'],
    }
```

El print va entre paréntesis ya que estamos usando Python 3 y karlita Python 2. Ella lo usará sin paréntesis.

```
<QuerySet [<Registrado: palotes@gmail.com>, <Registrado: test@gmail.com>, <Registrado: test@gmail.com>, <Registrado: test2@gmail.com>, <Registrado: email@email.com>, <Registrado: validacion@eduardo.com>, <Registrado: validacion@email.edu>, <Registrado: hola@test.edu>, <Registrado: hola@test.edu>, <Registrado: hola@test.edu>, <Registrado: usuario@usuario.edu>, <Registrado: juanito@alpargata.edu>, <Registrado: juanito@alpargata.edu>]>
[25/Apr/2021 11:20:46] "GET / HTTP/1.1" 200 3956
```

Si iniciamos sesión con nuestro superusuario y abrimos el terminal se nos mostrará todos los objetos registrados en nuestra base. Como podemos ver, está dispuesto en forma de lista. Por ello también podemos hacer que se liste cada instance:

```
39     if request.user.is_authenticated and request.user.is_staff:
40         for instance in Registrado.objects.all():
41             print (instance)
42             context = {
43                 "quererset": ['hola', 'adios'].
```

Recargamos...

```
Quit the server with CONTROL-C.
palotes@gmail.com
test@gmail.com
test@gmail.com
test2@gmail.com
email@email.com
validacion@eduardo.com
validacion@email.edu
hola@test.edu
hola@test.edu
hola@test.edu
usuario@usuario.edu
juanito@alpargata.edu
juanito@alpargata.edu
[25/Apr/2021 11:23:34] "GET / HTTP/1.1" 200 3956
[25/Apr/2021 11:23:34] "GET /media/js/bootstrap.js HTTP/1.1" 304 0
```

Y tendremos listados nuestros registros de uno en uno y separados. Si queremos saber cuantos registrados tenemos...

```
39     if request.user.is_authenticated and request.user.is_staff:
40         ide = 1
41         for instance in Registrado.objects.all():
42             print(ide)
43             ide += 1
44             print (instance)
```

Así cada uno tendrá un id:

```
test@gmail.com
4
test2@gmail.com
5
email@email.com
6
validacion@eduardo.com
7
validacion@email.edu
8
hola@test.edu
9
hola@test.edu
10
hola@test.edu
11
hola@test.edu
12
usuario@usuario.edu
13
juanito@alpargata.edu
juanito@alpargata.edu
[25/Apr/2021 11:28:29] "GET / HTTP/1.1" 200 3956
[25/Apr/2021 11:28:29] "GET /media/js/bootstrap.js HTTP/1.1" 304 0
```

También podemos hacer consulta de los atributos de cada objeto(nombre, email y timestamp):

```
for instance in Registrado.objects.all():
    print(ide)
    ide += 1
    print (instance.nombre)
```

Si queremos el nombre, basta con añadirlo como atributo:

```
None
4
test2
5
None
6
None
7
None
8
hola
9
hola
10
hola
11
12
Usuario
13
juanito
13
Usuario
```

Y finalmente, vamos a usarlo en nuestra plantilla. Para ello creamos una variable que contenga todos los objetos y lo añadimos a nuestro contexto:

```
if request.user.is_authenticated and request.user.is_staff:
    queryset = Registrado.objects.all()
    context = {
        "queryset": queryset,
    }
```

Recargamos...

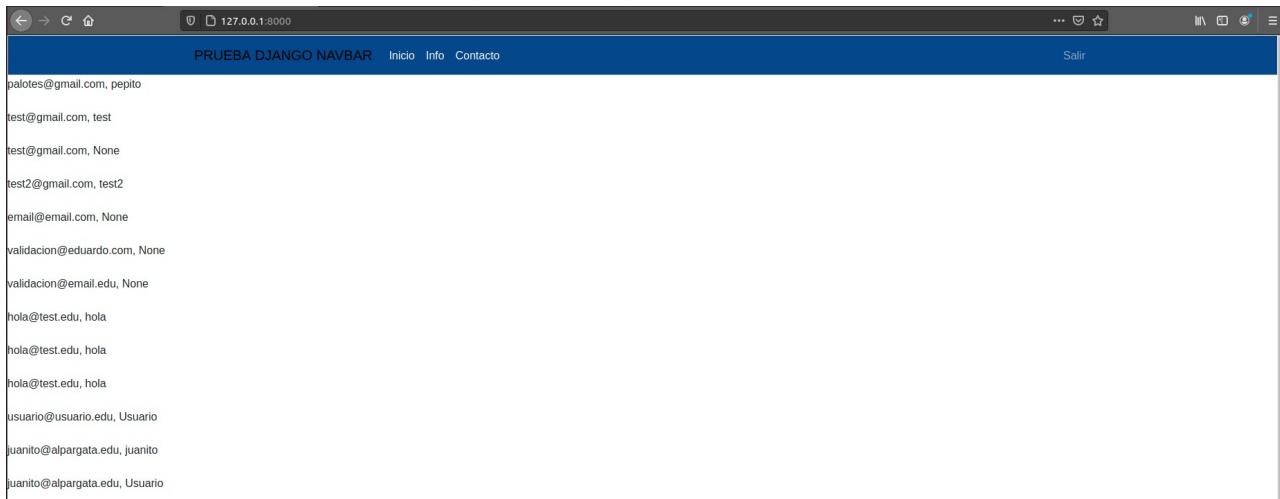


Y podemos ver todos los registros!

Una vez incluido el *queryset*, podemos manipularlo desde nuestra plantilla de la siguiente manera:

```
22  {% if request.user.is_staff %}
23  {% for instance in queryset %}
24  {{ instance }}, {{ instance.nombre }}<br></br>
25  {% endfor %}
26  {% else %}
```

Si quisiéramos dejarlo así este sería es resultado:



Vamos a mejorar la vista añadiendo una tabla:

```
22  {% if request.user.is_staff %} 
23  <table class="table">
24  {% for instance in queryset %}
25  <tr><td>{{ instance }}, {{ instance.nombre }}</td></tr><br><br>
26  {% endfor %}
27  {% else %}
28
```



También lo podemos meter en otra columna separada:

```

22  {% if request.user.is_staff %}
23  <table class="table" style="text-align: center;">
24  <tr><td>CORREO</td><td>NOMBRE</td></tr>
25  {% for instance in queryset %}
26  <tr><td>{{ instance }}</td><td>{{ instance.nombre }}</td></tr><br><br>
27  {% endfor %}
28  {% else %}
29

```

CORREO	NOMBRE
palotes@gmail.com	pepito
test@gmail.com	test
test@gmail.com	None
test2@gmail.com	test2
email@email.com	None
validacion@eduardo.com	None
validacion@email.edu	None
hola@test.edu	hola
hola@test.edu	hola
hola@test.edu	hola
usuario@usuario.edu	Usuario
juanito@alpargata.edu	juanito
juanito@alpargata.edu	Usuario

Y añadir la fecha:

```

22  {% if request.user.is_staff %}
23  <table class="table" style="text-align: center;">
24  <tr><td>CORREO</td><td>NOMBRE</td><td>FECHA</td></tr>
25  {% for instance in queryset %}
26  <tr><td>{{ instance }}</td><td>{{ instance.nombre }}</td><td>{{ instance.timestamp }}</td></tr><br><br>
27  {% endfor %}
28  {% else %}
29

```

CORREO	NOMBRE	FECHA
palotes@gmail.com	pepito	9 de Abril de 2021 a las 10:39
test@gmail.com	test	9 de Abril de 2021 a las 10:50
test@gmail.com	None	11 de Abril de 2021 a las 10:43
test2@gmail.com	test2	11 de Abril de 2021 a las 10:51
email@email.com	None	13 de Abril de 2021 a las 08:52
validacion@eduardo.com	None	13 de Abril de 2021 a las 08:57
validacion@email.edu	None	13 de Abril de 2021 a las 09:01
hola@test.edu	hola	13 de Abril de 2021 a las 18:06
hola@test.edu	hola	13 de Abril de 2021 a las 18:06
hola@test.edu	hola	13 de Abril de 2021 a las 18:09
usuario@usuario.edu	Usuario	13 de Abril de 2021 a las 18:09
juanito@alpargata.edu	juanito	13 de Abril de 2021 a las 18:17
juanito@alpargata.edu	Usuario	13 de Abril de 2021 a las 18:17

Están ordenados de más antiguos a mas recientes. Si creáramos uno nuevo se añade abajo del todo...

juanito@alpargata.edu	juanito	13 de Abril de 2021 a las 18:17
juanito@alpargata.edu	Usuario	13 de Abril de 2021 a las 18:17
nuevo@gmail.edu	nuevo	25 de Abril de 2021 a las 11:57

Si quisieramos cambiar el orden de aparición por fecha basta con añadir un ‘order by’ en nuestra vista donde se declara la variable:

```
if request.user.is_authenticated and request.user.is_staff:
    queryset = Registrado.objects.all().order_by("-timestamp")
    context = {
```

Y veremos como nos cambia el orden y el nuevo usuario está arriba del todo:

CORREO	NOMBRE	FECHA
nuevo@gmail.edu	nuevo	25 de Abril de 2021 a las 11:57
juanito@alpargata.edu	Usuario	13 de Abril de 2021 a las 18:17
juanito@alpargata.edu	juanito	13 de Abril de 2021 a las 18:17

Si añadimos el filtro ‘timesince’ al timestamp nos devolverá en la tabla el tiempo que ha pasado desde su creación:

```
{% for instance in queryset %}
<tr><td>{{ instance }}</td><td>{{ instance.nombre }}</td><td>{{ instance.timestamp|timesince }}</td></tr><br>
{% endfor %}
{% else %}
```

CORREO	NOMBRE	FECHA
nuevo@gmail.edu	nuevo	6 minutos
juanito@alpargata.edu	Usuario	1 semana, 4 días
juanito@alpargata.edu	juanito	1 semana, 4 días
usuario@usuario.edu	Usuario	1 semana, 4 días
hola@test.edu	hola	1 semana, 4 días
hola@test.edu	hola	1 semana, 4 días
hola@test.edu	hola	1 semana, 4 días
validacion@email.edu	None	1 semana, 5 días
validacion@eduardo.com	None	1 semana, 5 días
email@email.com	None	1 semana, 5 días
test2@gmail.com	test2	2 semanas
test@gmail.com	None	2 semanas
test@gmail.com	test	2 semanas, 2 días
palotes@gmail.com	pepito	2 semanas, 2 días

Además, podemos añadir filtros en el queryset de la siguiente manera (en nuestro views):

```
39     if request.user.is_authenticated and request.user.is_staff:
40         queryset = Registrado.objects.all().order_by("-timestamp").filter(nombre__icontains="o")
41         context = {
```

Y así podremos ver todos los usuarios que contengan en su nombre al menos una o:

CORREO	NOMBRE	FECHA
nuevo@gmail.edu	nuevo	hace 10 minutos
juanito@alpargata.edu	Usuario	hace 1 semana, 4 días
juanito@alpargata.edu	juanito	hace 1 semana, 4 días
usuario@usuario.edu	Usuario	hace 1 semana, 4 días
hola@test.edu	hola	hace 1 semana, 4 días
hola@test.edu	hola	hace 1 semana, 4 días
hola@test.edu	hola	hace 1 semana, 4 días
palotes@gmail.com	pepito	hace 2 semanas, 2 días

Lo mismo podemos hacer para *email*. Vamos a hacerlo con ‘*gmail*’, cambiando nombre por email y añadir como filtro la palabra:

CORREO	NOMBRE	FECHA
nuevo@gmail.edu	nuevo	hace 12 minutos
test2@gmail.com	test2	hace 2 semanas
test@gmail.com	None	hace 2 semanas
test@gmail.com	test	hace 2 semanas, 2 días
palotes@gmail.com	pepito	hace 2 semanas, 2 días

No solo podemos poner como filtro que contenga, sino que sea exactamente lo que pone en el filtro:

```
if request.user.is_authenticated and request.user.is_staff:
    queryset = Registrado.objects.all().order_by("-timestamp").filter(nombre__iexact="test")
    context = {
        "queryset": queryset
    }
```

CORREO	NOMBRE	FECHA
test@gmail.com	test	hace 2 semanas, 2 días

Y para terminar, vamos a añadir un contador a cada uno:

```
22  {% if request.user.is_staff %}
23  <table class="table" style="text-align: center;">
24  <tr><td>CONTADOR</td><td>CORREO</td><td>NOMBRE</td><td>FECHA</td></tr>
25  {% for instance in queryset %}
26  <tr><td>{{ forloop.counter }}</td><td>{{ instance }}</td><td>{{ instance.nombre }}</td><td>hace {{ instance.timestamp|timesince }}</td></tr><br><br>
27  {% endfor %}
28  {% else %}
29  
```

CONTADOR	CORREO	NOMBRE	FECHA
1	nuevo@gmail.edu	nuevo	hace 47 minutos
2	juanito@alpargata.edu	Usuario	hace 1 semana, 4 dias
3	juanito@alpargata.edu	juanito	hace 1 semana, 4 dias
4	usuario@usuario.edu	Usuario	hace 1 semana, 4 dias
5	hola@test.edu	hola	hace 1 semana, 4 dias
6	hola@test.edu	hola	hace 1 semana, 4 dias
7	hola@test.edu	hola	hace 1 semana, 4 dias
8	validacion@email.edu	None	hace 1 semana, 5 dias
9	validacion@eduardo.com	None	hace 1 semana, 5 dias
10	email@email.com	None	hace 1 semana, 5 dias
11	test2@gmail.com	test2	hace 2 semanas
12	test@gmail.com	None	hace 2 semanas
13	test@gmail.com	test	hace 2 semanas, 2 dias
14	palotes@gmail.com	pepito	hace 2 semanas, 2 dias

Y con esto, damos por concluido este video y esto curso!

CONCLUSIÓN

El curso ha concluido y, gracias al mismo, he conseguido elaborar de forma simple mi primera aplicación con **Django**. A partir de aquí le puedo dar la forma que quiera. Los *commits* hasta el 31 incluye la subida (no todos son por cada capítulo ya que he tenido que subir en algún que otro momento algunos vídeos con pruebas de vida de golpe). Esto no acaba aquí ya que subiré algún que otro *commit* más donde ajustaré detalles de la aplicación para que quede más presentable ya que esto no es más que un curso para aprender y son todo para hacer pruebas de como se haría.

Aquí concluyo con mi enlace al repositorio de [Github](#).